COMPUTER-AIDED MASK LAYOUT DESIGN FOR BIPOLAR INTEGRATED CIRCUITS

A Thesis Submitted
in Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

by

BIRADAR LINGANAGOUDA SHIVARAYAGOUDA

to the

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JANUARY, 1986

CERTIFICATE

Certified that the thesis entitled 'COMPUTER-AIDED MASK LAYOUT DESIGN FOR BIPOLAR INTEGRATED CIRCUITS' by Shri Biradar Linganagouda Shivarayagouda has been carried out under our supervision and this has not been submitted elsewhere for a degree.

(Dr. S.G. Dhande)
Professor
Department of Mechanical Engg.
and Computer Science and Engg.
Indian Institute of Technology
Kanpur.

(Dr. M.M. Hasan)
Professor
Department of Electrical Engg.
Indian Institute of Technology
Kanpur.

January, 1986

ACKNOWLEDGEMENTS

I would like to express my heartfelt thanks to Dr. M.M. Hasan and Dr. S.G. Dhande, my thesis supervisors, for their guidance and encouragement throughout this work. I have benefitted immensely from the discussion I had with them.

I would like to thank my friend Mr. M. Sethuraman, with whom I had quite a lot of discussion about my thesis. My thanks are to my friend Mr. S. Dutta who took pains in proof reading. Last but not the least, thanks are due to Mr. C.M. Abraham for the excellent typing of the thesis.

Biradar Linganagouda S.

Contents

List of Figures

List of Tables

SYNOPSIS

The fabrication of single crystal silicon monolithic bipolar integrated circuit involves several process steps which demand the preparation of precise artwork. Some of them are diffusion, growth of epitaxial layer, oxidation, and metallisation. Once a circuit diagram is decided, a detailed layout diagram is needed before starting the fabrication process. From this detailed master layout diagram, individual masks for each process step has to be prepared. The characteristics of circuit elements depends upon their geometry. Before going for fabrication of such a circuit, it is required to prepare a layout diagram of the circuit. The initial artwork, from which photo masks are made, is highly an accurate drawing scaled between 100 and 1000 times final size. If the circuit is simple with less number of components, one can do the artwork by hand and get the accurate masks, but for larger circuits manual artwork is prone to mistakes. Thus, at this stage one can think of design automation for getting the circuit layout, i.e., master layout and masks. A digital computer with graphic terminal facilitates the design automation process.

Here a software program has been developed to get the master layout as well as layouts of different layers separately by giving circuit description of any required basic bipolar integrated circuit. The program has got autorouting feature, which finds

the minimum distance path without crossovers. The developed layout generator can be sent for mask preparation. This mask is used during lithographic process for transferring the pattern on the silicon surface. When a full-custom design approach is too time-consuming or not justifiable financially due to low production volume, then the semi-custom design approaches are a compromise solution. Currently the semi-custom design approaches utilize gate arrays. Gate arrays of ECL and TTL have been extensively used in main frames and mini-computers. The developed software is capable of giving layouts of gate arrays.

One can extend the software program so as to enable the basic building blocks to form SSI and MSI. The package can be further extended to include layout of LSI circuits. The problem the designer would face is that of the size of the display screen and the plotter to get the layouts of 100 to 1000 times final size.

CHAPTER 1

INTRODUCTION

In a few years, the field of integrated circuits has experienced explosive growth. The integrated circuit is a miniature, low cost electronic component that performs a high level function. The circuit elements-transistor, diode, resistor and capacitor are made at appropriate places in a semiconductor substrate and metalisation is done for inter-connection. The fabrication of integrated circuits involves several process steps which demand the preparation of precise artwork. The characteristics of circuit elements depends upon their geometry. After the circuit design to achieve the re-quired performance, the designed gogeometry of circuit elements is used. The geometrical pattern of all circuit elements along-with their metallisation is called as the layout of the circuit. The initial artwork, from which photomasks are prepared, is highly an accurate drawing scaled between 100 and 1000 times final size. For simpler circuits one can do the artwork by hand to such an accuracy. But for complex circuits the hand artworking is prone to mistakes. The use of computer aided design techniques have led to an improvement in the accuracy and a reduction in the turnaround time of the artwork. Thus it has been thought of design automation of layout. This has made a great impact on LSI and semi custom-made ICs. There are

usually a very large number of ICs on the same wafer. The wafer mask is a matrix of identical circuit patterns generated by repetitive stepping of the appropriate mask master. This mask stepping is universally done by automatic equipment like a step and repeat camera.

1.1 STATEMENT OF THE PROBLEM

The aim is to develop a software package which draws layout drawing with nonautoplacement and autorouting features. It has been resorted to nonautoplacement because for autoplacement the compution time required is quite large. Not only this the layout obtained by autoplacement will not be satisfactory in the sense that the total substrate area used will be about 20% more than that required by nonautoplacement. The package is capable of giving master layout as well as the layouts of different layers separately for mask preparation.

1.2 EVOLUTION OF INTEGRATED CIRCUITS

Semiconductors were used in electronics long before the invention of the transistors in 1947. The first radio detectors were point contact diodes, in which a metal point made contact with galena or silicon [1]. The period, 1947 to 1960 is called the transistor era. Since 1958 till today is the era of integrated circuits. To begin with, it was a SSI circuit, which has very few transistors. Then the level of complexity increased in the order SSI, MSI, LSI and VLSI circuits.

LSI digital IC consists of several gates and flip flops. An MSI digital IC is a device that performs a high level logic function and has a minimum of 10 gates and a maximum of 100 gates. LSI devices in the bipolar processing class are not many. But at LSI and VLSI level, MOS technology is popular because of their inherent advantages. The present day complex VLSI circuit contains 400000 transistors or more. New bipolar technology notably Integrated Injection logic, seriously challenge the dominance of MOS in the LSI field. The requirement for a high operating speed in a complex IC has pushed-up the development of Integrated Injection logic [3].

1.3 CLASSIFICATION OF INTEGRATED CIRCUITS

The two general fabrication categories of ICs are the monolithic and hybrid types [3]. The greater number of devices sold today are bipolar and unipolar monolithic ICs. Bipolar ICs are those devices whose components are current controlled and require both positive and negative polarity charge carriers for operation in their active elements. Unipolar monolithic ICs pertain to those devices whose components are primarily voltage controlled and have a single polarity operational charge carriers in its active elements. Thus hybrid, bipolar and unipolar monolithic forms the classification of integrated circuits according to fabrication.

The another classification of integrated circuits is by function. Digital ICs are those devices whose inputs and outputs

have two discrete states. Linear ICs are those devices whose inputs and outputs are proportionally or mathematically related.

The Digital ICs can inturn be classified according to their scale of integration as shown in Figure 1.

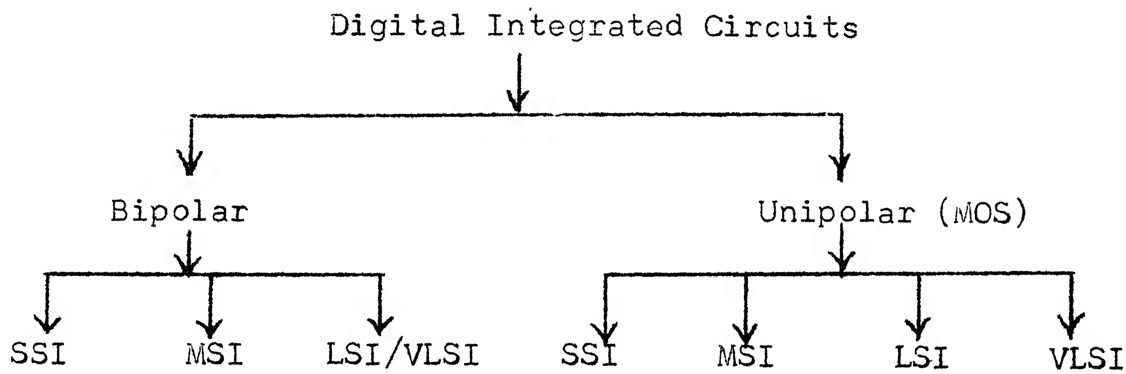


Figure 1 . Classification of Digital Integrated Circuits

1.4 ADVANTAGES OF AUTOMATED SYSTEMS

1. The prime motive behind automated mask generation is to reduce design time and cost of the final product. Its need is specially felt in making modifications in a previous design which would otherwise require much larger turnaround time.
2. Automatic drafting equipment exceeds the manual accuracies and allows working at much smaller scale factors. This results in less time, less cost and less photographic distortion in the subsequent reduction.

3. By the use of automated systems many manual errors and omissions which normally defy visual inspection are avoided.

4. The speed of automated system increases the capacity of a designer to handle custom made IC designs and the high precision now offered, allows the generation of artwork for systems, which were so far considered too complex to be handled by manual techniques.

CHAPTER 2

BIPOLAR INTEGRATED CIRCUITS

The integrated circuit (IC) is the result of a multistep physical and chemical processes. The manufacture of ICs is an extremely complex process. Although the process is complex and as a whole costly, the net result is miniature circuits that are inexpensive and reliable.

2.1 BIPOLAR INTEGRATED CIRCUIT TECHNOLOGY

The fabrication of a single-crystal monolithic circuit consists basically of constructing diodes, transistors, resistors and capacitors on a single substrate and of providing sufficient isolation to minimise the parasitic interaction between constituent parts of the total circuit. The fabrication process is a method of arranging the basic materials in a geometric pattern which produces the desired electrical behaviour. The technology consists of proper sequence of four basic processes [2].

1. <u>Photoresist and oxide masking</u>

The diffusion constant of typical impurities is much smaller in silicon dioxide than in silicon. A thin layer of silicon dioxide can be used on a silicon wafer to selectively control areas in which diffusion of impurities takes place. In the areas where diffusion into the silicon is desired, the layer of silicon

dioxide is removed by etching. A photolithographic technique is employed to implement the etching of apertures in silicon dioxide with the precision required for ICs. The surface of silicon dioxide is uniformly coated with photoresist. A glass mask containing a pattern of opaque and transparent region is placed over it and exposed to ultraviolet light. This light polymerises the exposed portion, and the unexposed photoresist and silicon dioxide will be removed by chemical solvents. Then the windows will be ready for diffusion process.

2. Impurity Diffusion into Silicon

In the diffusion of impurities into silicon two types of boundary conditions are introduced. The first one is constant-source diffusion, where the wafer is placed in a furnace at a temperature of $1100^{o}C$ and is exposed to a vapor containing a compound with the desired impurity atoms. In this the diffusion profile is complementary error-function. The second boundary condition is the limited source diffusion, in which a fixed number of impurity atoms/cm^2 is deposited on the silicon surface. In the second condition the diffusion profile is gaussian distribution. The characteristic of both the diffusion techniques is that the concentration of impurity atoms varies with depth into the silicon.

3. Epitaxial Growth of Silicon Containing Impurities

It is often desirable to have more flexibility with regard

to impurity distribution than is provided by diffusion techniques. A constant impurity distribution may be desired, or it may be necessary to avoid the problems associated with more than two diffusion cycles. This is accomplished by epitaxial growth. In this process, silicon wafers are placed in a furnace at about 1200°C. A vapour containing H_2 and $SiCl_4$ is passed over the wafer, and due to chemical reaction HCl is freed and elemental silicon is deposited on the wafer. Epitaxial process is generally used only where nonlocalised junctions are required.

4. Metallisation

In order to provide low-resistance interconnections among the various components which are fabricated on the substrate, a metal evaporating technique is used. The area of silicon which are to be contacted are exposed by the photo-resist process,and the wafer is placed in a vacuum system in which the pressure is reduced to approximately 10^{-6} millimeter of mercury. When a layer approximately 1 micron thick has been deposited, the wafer is removed and the metal is etched from all the portions except the desired portion by a photo-resist process. Fine metal wires may be bounded now to appropriate places on the wafer, enabling connection to external circuits.

The fabrication of an IC requires from five to as many as fifteendifferent masks. The typical sequence of processes are burried layer diffusion, epitaxial layer and oxidation, isolation diffusion,base diffusion, emitter diffusion capacitor oxidation, and metallisation.

2.2 COMPONENTS OF BIPOLAR INTEGRATED CIRCUIT

The designer of discrete circuits has a large number and wide variety of components that he can use. He can select from resistors, inductors and capacitors to transformers, switches and indicators. The designer of monolithic circuit is not so fortunate. He has a restricted number of components that he can use, and their performance and rating is process limited. The components that may be integrated are transistors, diodes, resistors, and capacitors. Capacitors are costly to integrate and are used sparingly [2],[3].

Transistor

Most widely used three terminal device is the npn transistor. Its overall performance is superior to that of its complement, the pnp transistor. The npn transistor is less difficult to manufacture. There are many geometries of transistor. One among them in the top view (i.e., the layout schematic) is as shown in Figure 2.1.

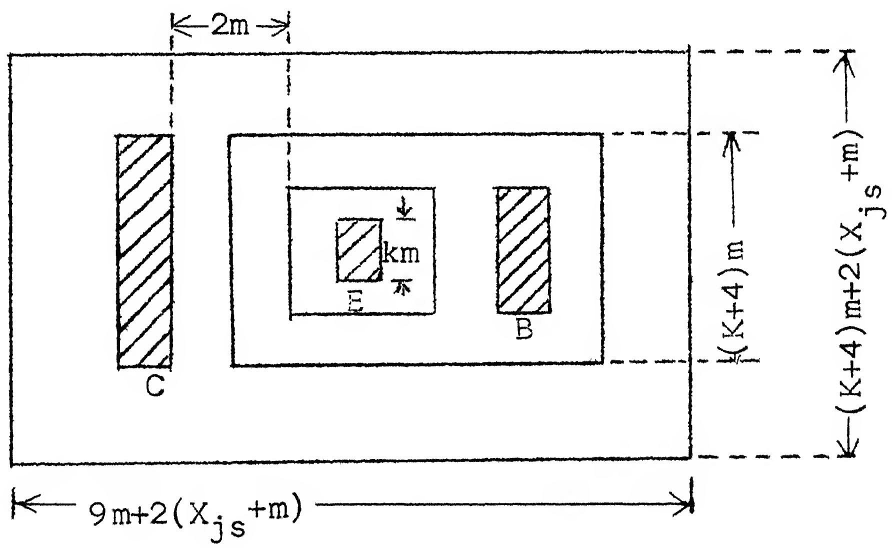


Fig. 2.1 Top view of an npn transistor

In the figure 2.1 m,k and X_{js} are design parameters. The typical values are 10 microns, 2, and 10 microns respectively. In the technology these typical values for m,k and X_{js} are selected.

Diode

A solid state diode is formed by making an n diffusion into a p wafer (or vice versa). The n area is the diode's cathode, and the P area is the anode. Diode action occurs at the pn junction. As the transistor has got emitter and collector junctions, we can make use of any one junction as a diode, leaving the third terminal free or appropriately shorting. The top view of a diode is as shown in Figure 2.2. The design parameters m and k are same as those for transistor.

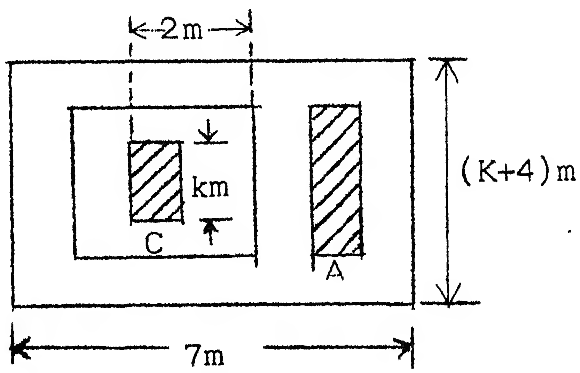
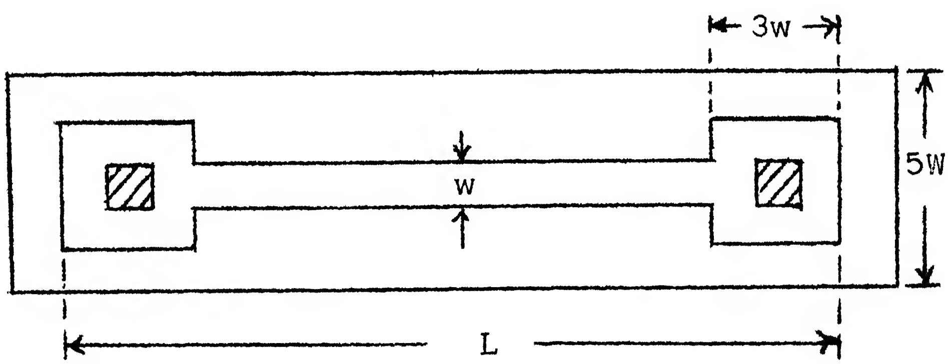


Figure 2.2 Top view of a diode

Resistor

Diffused resistors are usually formed during the P-type base diffusion, for small values of resistance the emitter

diffusion is used. Diffused resistors have a manufacturing tolerance of about 20% with respect to absolute values, while ratio of resistors may be held within 5%. There are many geometries for resistances. If the resistance value is small the geometry shown in Figure 2.3a is used, for larger valued resistance then that of Figure 2.3b is used. 'R_s' is the sheet resistance and typically 200 Ohms/sq for base diffusion and 2.5 Ohms/sq for emitter diffusion. 'W' is the width of resistance and typical value is 10 microns. For the software developed the value of 'm' of transistor and 'W' of resistor should be same. The technology makes use of 200 Ohms/sq and 10 micros as the values for R_s and w respectively. In this selected technology the software cannot give correct layout for resistances of value less than 400 Ohms. Under such circumstances the parallel resistances are to be considered to realise a small value resistance.



$$R = (L-6W) \times \frac{R_s}{W} + 1.3\, R_s$$

Figure 2.3a Top view of a small valued resistor

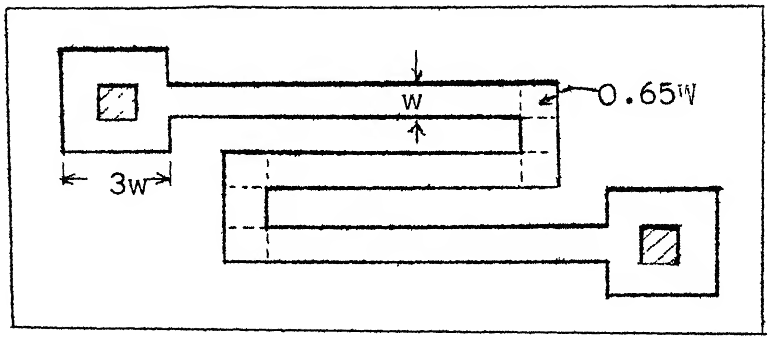


Figure 2.3b Top view of a large valued resistor

2.3 LAYOUT OF BIPOLAR INTEGRATED CIRCUIT

The layout of circuit is necessary for fabrication by monolithic integrated circuit technology. The designer has to use the geometrical shapes for components as discussed above and appropriately place them, so that the interconnecting metallisation will not be longer. While preparing a layout the designer has to follow the following guidelines [2].

- Within the limits of minimum spacing, minimise the area so as to obtain low capacitance and high yield.
- Group components together wherever possible to minimise the number of isolated regions required.
- In single crystal circuits, large area of isolation junction should be avoided because of the attendent large leakage currents.
- The number of interconnection crossovers should be minimised, because one crossover consists of a heavily doped n^+ region below the oxide.

5. To ensure better temperature tracking among the circuit elements, the layout should be arranged to maintain as nearly as possible uniform power dissipation in the chip.

6. The layout should be arranged in such a way, so that connections can be made to the substrate and the isolation regions. This permits proper biasing of these regions to minimise parasitic effects.

7. The layout should be arranged in a way so that the bonding pads are at the periphery and evenly distributed all along the periphery.

8. The crossovers of metallisation can be avoided by passing the metallisation run over the resistors.

Apart from these guidelines the designer must follow the following layout specifications [1]. For LSI circuits, the device size will reduce.

1. Minimum feature size on mask is 8x8 micron^2.
2. Minimum width of metal line is 10 microns.
3. Minimum metal to metal spacing is 10 microns. But for very short runs it is 8 microns.
4. Minimum worst case spacing is 1 micron.
5. Minimum pad size is 100x100 micron^2.

   Minimum spacing between

6. burried layer and isolation is 17 microns.
7. deep collector and isolation is 18 microns.

8. base diffusion and isolation is 16 microns.
9. deep collector and base diffusion is 16 microns.
10. emitter diffusion to base diffusion is 6 microns.
11. emitter diffusion to emitter contact is 5 microns.
12. emitter diffusion to base contact is 7 microns.
13. base diffusion to base contact is 5 microns.
14. pads is 100 microns.

2.4 MASKING AND ITS ECONOMICS

Each circuit requires a number of masks usually between 5 and 15 depending upon the number of processing steps. In general masks are needed for each diffusion, oxide opening and etching. The various layers are drawn very accurately with a high precision in their relative locations. The following are some of masks needed for a simple circuit.

1. Burried layer diffusion mask.
2. Isolation region selection mask.
3. Base diffusion mask.
4. Emitter diffusion mask.
5. Capacitor location selection mask.
6. Oxide opening for ohmic contact mask.
7. Metal etching selection mask.

In an industrial set up the entire process of converting the circuit schematic into masks takes from 80 to 320 man hours, depending on the complexity of the circuit. The artwork

generation process is such that the next step can not be carried out without the successful completion of the previous step. The turnaround time is high and no reduction is possible by employing more persons. To reduce the preparation time and cost, an automated system is considered essential for a large manufacturing firm.

CHAPTER 3

SOFTWARE DEVELOPMENT FOR IC LAYOUT

The literature pertaining to the field of electronic circuits and electromagnetic components, are concentrating mainly on computer aided design these days [7]. The expense of developing a circuit layout program must be justified in comparison with that for manual design. The justification for large system design computer programs must come from the savings in time and in the improved cost-effectiveness of a better design.

3.1 COMPUTER-AIDED DESIGN AND ITS APPLICABILITY

Transistor circuits require greater attention to component arrangement because of the difficulty of routing printed interconnections without metallisation cross-overs. However, integrated circuits (ICs) present more complex layout design problems. Computer can be used for such a complex layout design and for interactive correction. Low cost alongwith reduced time required for the development of new integrated circuits could definitely popularise computer for its wide-spread use. One way of achieving a substantial speed up in design time is to use a computer with graphical display and on-line interaction with the designer [8].

Computer-aided design as practiced today is defined as the replacement of the difficult and tedious manual computations in the design process with a computing machine [7]. There are certain characteristics of a design problem that make computer-aided design applicable. One or more of the following are appropriate.

1. Can the problem be programmed for a rapid solution?
2. The design is interactive in nature.
3. The number of variables is high so that a manual solution is difficult.
4. The computations are lengthy and time-consuming.
5. There are fallout benefits derived as for example, information is obtained about the end product without its actual fabrication, which may be expensive.

3.2 FLOW CHART OF DEVELOPED SOFTWARE

The flow chart of the developed software is as shown in Fig. 3.1. The block 1 reads the 'input' data file and prepares the necessary data of circuit elements to draw the layout. Block 2 draws the isolation region. Block 3 draws the base diffusion. Block 4 draws the emitter diffusion. Block 5 draws the oxide cut for contacts. Block 6 draws the pads and generates their addresses. Block 7 initialise the routing matrix, generates the addresses of terminals of components, and routes between the interconnecting terminals. You can refer the figure 3.1 for details of these blocks.

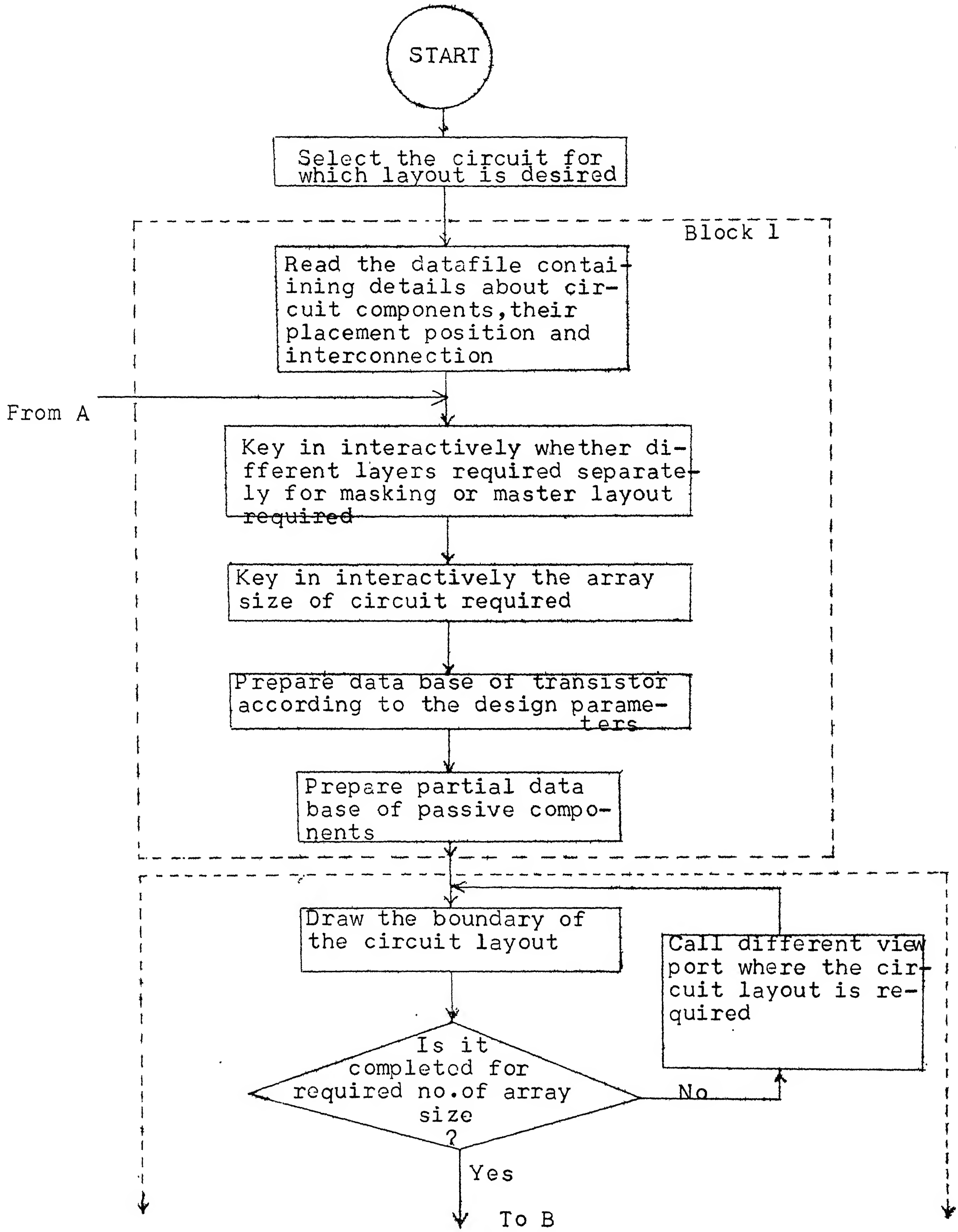


Fig. 3.1 Flow chart of developed Software

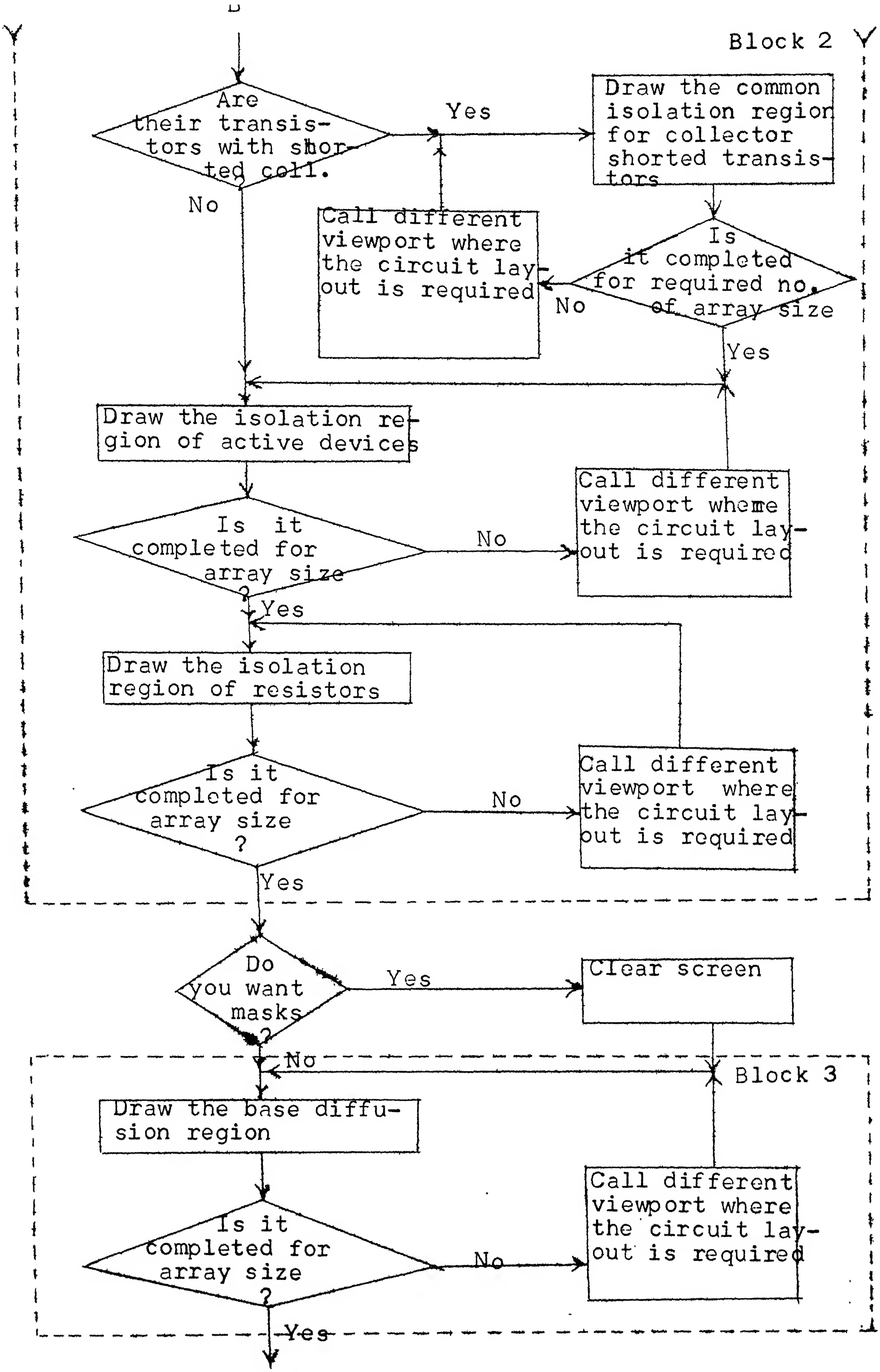


Fig. 3.1 Flow chart developed Software (contd.)

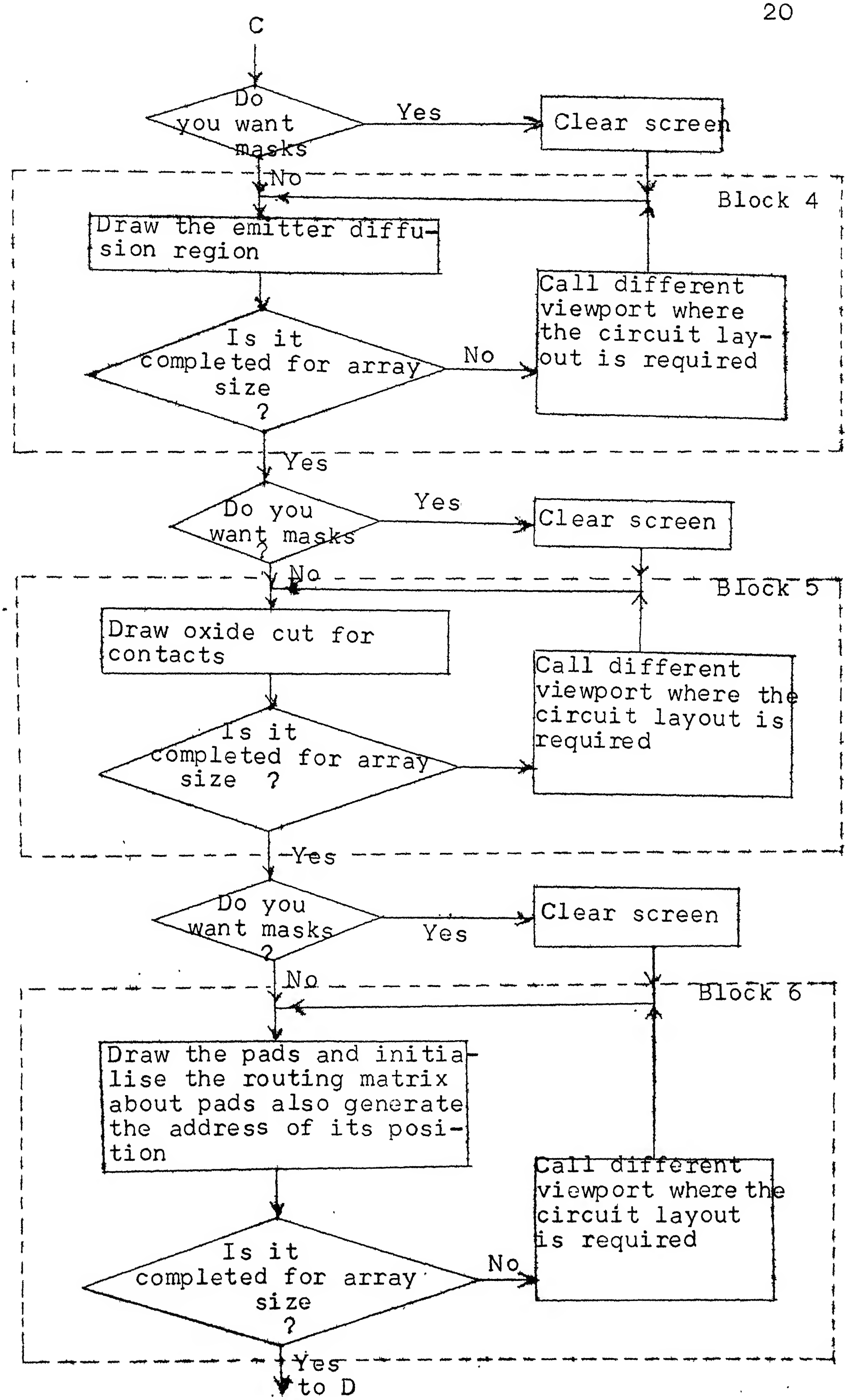


Fig. 3.1 Flow chart of developed Software (contd...)

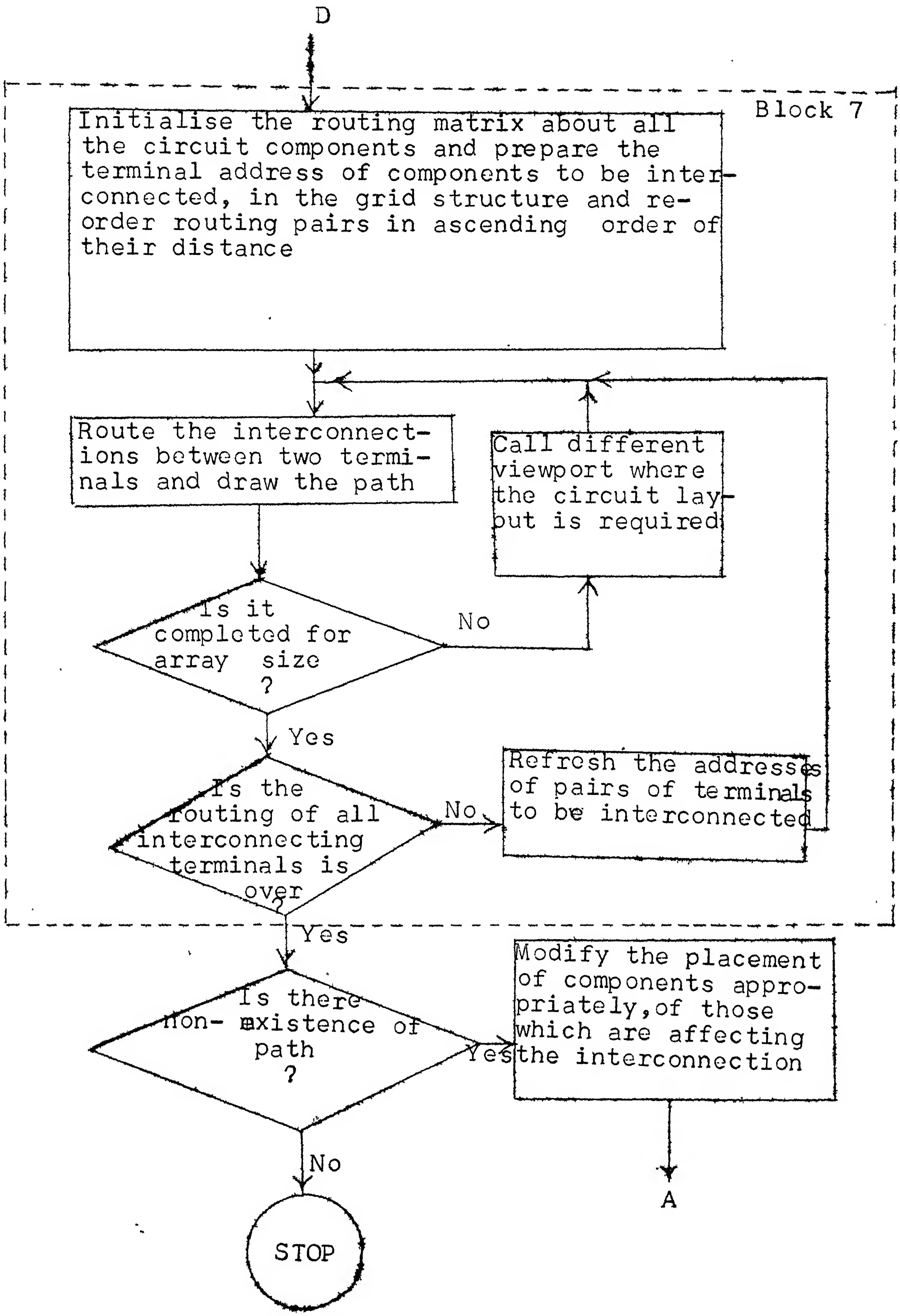


Fig. 3.1 Flow chart of developed Software

3.3 AUTOROUTING ALGORITHMS

The central problem in computer-aided layout design is of formation of conductor interconnection pattern for the given circuit configuration. To achieve this automated interconnection there exists many autorating algorithms. Rat in a maze and Lee path connection algorithm are two among them. These approaches maintain a two dimensional model of the working area. These algorithms are most widely used for finding wire paths on printed circuit boards. The same algorithms can also be used for finding metallisation paths in integrated circuit layouts.

3.3a Rat in a Maze

A stepping procedure has been made use of , in this so-called 'rat in a maze' problem for routing interconnections. The working area is divided by a fine grid and the obstacles are indicated in a preassigned code. The only information available to this hypothetical rat is the coordinates or the grid location of its destination. So it has the knowledge of the general direction in which it is supposed to move. This algorithm involves the following steps.

1. The rat is allowed to move only along the vertical and horizontal directions.
2. Move the rat step by step, and the step which reduce the distance between present location and the destination is positive and the move which increases the distance is negative.

3. When it is moving in a direction so as to reduce the distance between its destination and itself, if it encounters an obstacle turn the rat through 90° towards the destination, i.e., it goes from vertical direction to horizontal or vice versa.

4. If rat encounters obstacles in the forward direction, retrace back the rat by some steps or fully to the starting grid.

5. Then again start searching from some other direction.

As the rat is having a knowledge of direction it may take more time or some time fail to search the destination due to the obstracles on the path. That is why one can think of some other algorithm which ensures the search if a path exists with less amount of computation.

3.3b Lee Path Connection Algorithm

The Lee path connection algorithm is probably the most widely used method for finding wire paths [11],[13]. The Lee algorithm is based on expanding a wavefront one point to another. At each step, cells on a diamond-shaped front wave are expanded one step further. Each cell is marked with a trace back code. The trace back code stores the direction to the source of expansion. The algorithm has the following property :

1. It will always find a path if one exists and it is a rectilinear path (a path composed of horizontal and vertical line segments).
2. The path it finds will always have the minimum possible cost. Path cost is any measure the user wishes to minimise, and may include length, cross overs with existing wires (if permitted) and nearness to other wires.

The Lee algorithm requires a large memory for large and dense layouts. But the router guarantees a connection if it exists, regardless how complicated the grid may be. The speed improves as the area gets more congested.

In its original form, Lee's algorithm uses two lists, L, the list of frontier cells which are to be expanded and L1 the list of their neighbors. The principle iteration in the algorithm involves the following four steps.

<u>Algorithm 1 :</u>

1. For each admissible (i.e., non obstacle) neighbor of a cell in L, put in L1.
2. Adjoin to L all of those cells in L1. If any of these is the goal it is done.
3. Delete from L any cell whose neighbors have all been permanently labelled, and clear L1.
4. If L is empty no path exists. Otherwise repeat from Step 1.

As this algorithm takes more computation, so more time is required for search. Faster search has been resorted using Lee-Moore algorithm [15].

The basic technique by Moore and Lee, commonly referred as the Lee-Moore algorithm can be informally stated as follows. For simplicity of argument a single wiring plane is taken. The technique can be adapted in situations where multiple planes are involved. Let one of the points be called source and the other the destination. It has been made use of two lists of cells called list 1 and list 2 and 3 status markers per cell. One for the availability of cell, next whether the cell has been visited and lastly if so from which direction it is visited.

Algorithm 2 : Lee-Moore Algorithm

LM1. Initialisation:

Mark source cell as visited

List1 ← Source cell.

LM2. Propagation : starting with empty list 2,

For each cell in the list 1 :

new ← the neighbor cells of the current cell that are not permanently occupied (blocked) and not visited.

Mark new cells as visited and the direction visited from.

If destination $\in$ new, go to LM3, else append new to list 2.

Let List1 ← list2 (if list 2 is empty, the path does not exist)

Go to LM2.

LM3. Backtrace: Starting from the destination cell, follow the directions noted on the cells to the source cell.

This Lee-Moore algorithm is suitable for autorouting.

How exactly this Lee-Moore algorithm is implemented in this software development is discussed in next section.

3.4 SOFTWARE DESCRIPTION

The software developed is independent of the type of computer used. It can be implemented in any computer which has the facility of a graphic terminal and a plotter. The software has been written in FORTRAN 77 language and it makes use of PLOT 10 interactive graphics library routines. This software is implemented on OMEGA computer with a high resolution graphic terminal . Brief description about the PLOT 10 package is given in Appendix.

With software developed, circuit layout is performed automatically, subject to human review and modification. It results in automatic preparation of mask artwork, which defines the geometry of the circuit for manufacture. The software package performs the following functions.

1. Accepts an input circuit definition in a datafile, in a form similar to input for circuit analysis.
2. After the data have been validated, the location of components on the substrate is done manually.

3. Determines size and shape of every mask pattern for every component, from parameter value and technological specifications.
4. Displays initial layout with all components presented in true outline size and shape, arranged in the same relative positions as specified in the input data file.
5. If the displayed layout is alright it is stopped. Otherwise change of position of components has to be done. Layout modifications can also be used to solve interconnection problems.
6. Generates display patterns for all mask designs and master-layout.
7. Provides open-ended data structure to accommodate circuits of any desired size.
8. Provides layouts of gate arrays by making use of different viewports.

The software package needs seven blocks of data in an input data file. The block number in sequence, alongwith the type of data it represents, is given in Table 3.1.

If the circuit contains shorted collector transistors, then the software needs one more data file which should contain the coordinates of the rectangular region of common isolation island for all the transistors of this type.

Table 3.1

Structure of Datafile

Data blocks	Data types
Block 1	Technological specifications.
Block 2	Total number of components and their details such as code, terminal numbers, value, orientation, sense and positional co-ordinates.
Block 3	Total number of interconnections and the connecting terminal number pair.
Block 4	Total number of pads and their positional coordinates.
Block 5	The routing matrix (i.e., the grid) dimension.
Block 6	Number of coordinates which constitute the resistor island and their values.
Block 7	Number of coordinates which constitute the IC border and their values.

The codes which are used in this package for different components are as shown in Table 3.2.

For diode code 3 is used, where,in either the collector junction or the emitter junction is made use of appropriately during interconnection.

Each component of the circuit is identified as, composed of three terminals to make a common data structure for all the components. The third terminal of resistor is represented as -1 in

Table 3.2

Component Codes

Type of component	Code
Resistance	1
Transistor	2
Transistors with shorted collector	3
Multiemitter transistor	4

the data which does not have any meaning. In the datafile the three terminals of the transistor are identified in sequence as collector, base and emitter. As the terminals in multiemitter transistor are more the remaining emitters are identified in succession followed by the first emitter.

In the datafile the value of resistance should be entered in ohms. The meaning for the value of transistor is its number of emitters.

The next two types of data to be considered are orientation and sense of the component. The orientation tells whether the component layout is vertical or horizontal. The orientation can take the value either 0 or 1. '0' for horizontal and '1' for vertical. Sense tells whether the component layout is in normal form or inverted form. The normal and inverted

are same for resistor layout, because being symmetric about its centre. Whereas for transistor layout it will be different because not being symmetric about its centre. The type of layout and their orientation and sense code are given in Table 3.3.

The remaining types of data which have been used are explained very clearly in the enclosed program listings.

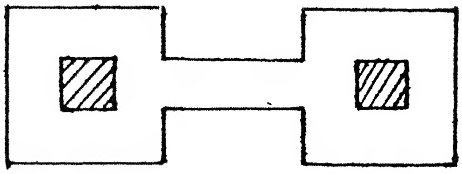
The Lee-Moore algorithm has been implemented for auto-routing. First of all let us consider the selection of data structure for router.

There are two basic data structures to describe the location of cells and interconnections. A segment data structure and a matrix data structure. Generally position resolution for cells or interconnections on systematic chips needs to be no greater than 1 part in 250. Thus a two dimensional matrix would require 62500 data locations to represent the chip. Since many modern chip technologies allow two or three levels of interconnection, 1 87500 locations are required.

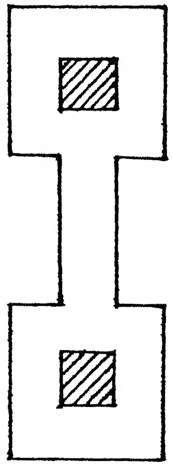
In the segment data structure, a list is made for each x and y coordinates. The segment data structure uses less core than the matrix approach, and the routing programs execute relatively quickly. But the programming using the segment approach is generally more complicated than for the matrix

Table 3.3 Orientation and sense of resistor and transistor

Resistor

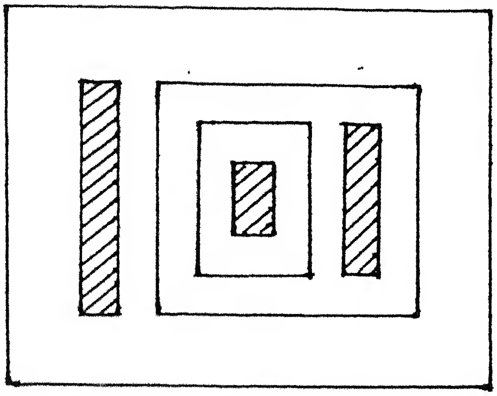


orientation=0
sense =±1

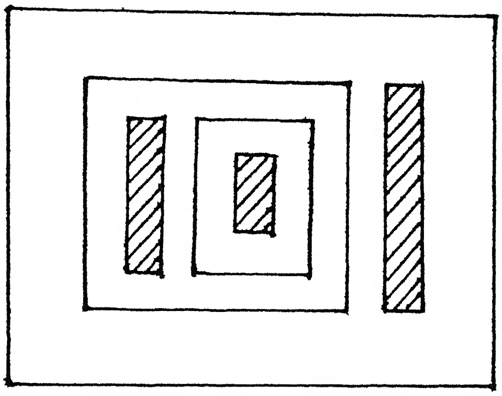


orientation = 1
sense = ±1

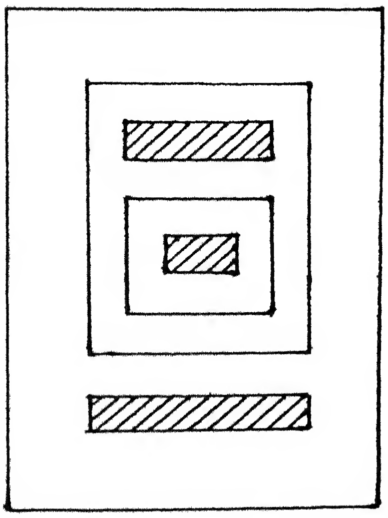
Transistor



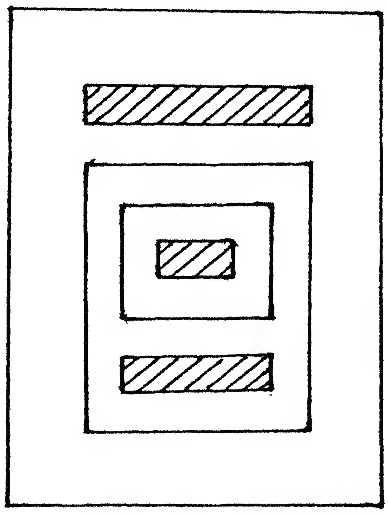
orientation = 0
sense = 1



orientation = 0
sense =-1



orientation = 1
sense=-1



orientation = 1
sense = 1

approach. So to implement the Lee-Moore algorithm the matrix data structure is used. Each cell of the grid structure is having a decimal number with eight digit positions in the format as shown below.

P	Z	Y	Y	Y	X	X	X

The most significant digit 'P' can take the value either O or 1. O implies that the cell is available for routing, whereas 1 implies the cell is not available for routing (i.e., permanently occupied). Next least significant digit 'Z' can take the value either O or 1. O implies that the cell has not been visited during the expansion of wavefronts, 1 implies that the cell has been visited during the expansion of wavefront from some other cell. Next three digits YYY can represent any number from OO1 to 999. This indicates the row number in the grid structure (i.e., matrix). Last three digits can also represent any number from OO1 to 999. This indicates the column number into grid structure. Thus in all YYY and XXX together will indicate the address of the cell from which the particular entered cell has been visited. This will help for easy retrace after finding the path. To start with the matrix is initialized with all zeros. Afterwords according to the component locations the appropriate cells have been filled with P digit as 1. Then the routing is tried. The

remaining thing is self explanatory if you go through the program listings. The software has got the capability of giving the layouts of the size, which can be embedded in a 999 x 999 matrix.

There are in all eleven subroutines present in the software. The purpose and explanation about them has been given in the program listings.

3.5 INSTRUCTIONS FOR USER

Prepare the datafile by the name 'input' and the data entry should be in the sequence of blocks as shown in Table 3.1. The READ statements which have been used to read the 'input' file are as under. And the type of the data i.e., integer or real is implicit from the name of the variable referred except for those which are declared explicitly.

```
      INTEGER TYPE, TERM1, TERM2, TERM3, ORINT, SENS, CONN,
 1    PADN
      REAL XJS,M,K,W,RS
      READ *, XJS,M,K,W,RS
      READ *, N
      READ *,(TYPE(I),TERM1(I),TERM2(I),TERM3(I),VAL(I),
 1    ORINT(I),SENS(I),POSI(I,1),POSI(I,2),I=1,N)
      READ *,N1
      READ *,(CONN(I,1),CONN(I,2),I=1,N1)
      READ *,N2
```

```
READ *,(PADN(I),PADPO(I,1),PADPO(I,2),I=1,N2)
READ *,N3
READ *,N4
READ *,(RILNDX(I),RILNDY(I),I=1,N4)
READ *,N5
READ *,(BORDRX(I),BORDRY(I),I=1,N5)
```

The ground pad should be entered at last in block 4 of datafile. For further understanding one can always refer the input files such as Tables 6.1 through 6.3 of Chapter 6.

If the circuit contains shorted collector transistors, then prepare another data file by the name 'inpu22' and the data entry is of single block with real type. The READ statement which reads this datafile is as shown below. For further details please refer the Table 6.4 of Chapter 6.

```
READ *,(STRAIX(I),STRAIY(I),I=1,5)
```

The value of N3 should not exceed the dimension of routing matrix and 999.

CHAPTER 4

BIPOLAR SMALL SCALE INTEGRATED CIRCUITS

The package is able to give the layout of any circuit. After developing any software package it is worth to make some testruns. It has been tried to test the package for small scale integrated circuits. Just for illustration one linear circuit the differential amplifier of the type (A3028 differential amplifier with collector resistances, two digital circuits the transistor-transistor logic gate and the emitter-coupled logic gate are considered.

4.1 DIFFERENTIAL AMPLIFIER

The linear group of integrated circuits (ICs) attained its importance and prominence as a group when it eventually became possible to provide operational amplifier at a reasonably low cost. The first stage of the operational amplifier, i.e., differential amplifier plays a vital role in achieving the present day operational amplifiers and it has got the following characteristics [2].

1. Have zero output with zero input
2. It provides constant gain without having variation with temperature, and input level.
3. It provides linear characteristic over a wide dynamic range of operation.

4. Low offset voltage and low noise
5. Low temperature drift and high common mode rejection ratio (CMRR)

Figure 4.1 is the circuit schematic of the differential amplifier with a constant current source of the type CA 3028 differential amplifier with collector resistances. In the schematic the terminal numbers of the components and the numbers of terminals which go to the outside world are indicated. This helps in preparing the input data record.

4.2 LOGIC GATES

Logic gates are digital circuits, which can be broadly classified into two categories. One is saturated logic and the other is unsaturated logic. Some of saturated logic gates are resistor-transistor logic (RTL), diode-transistor logic (DTL), and transistor-transistor logic (TTL). RTL, DTL and TTL have came up in the order of their speed of operation and fanout capability. Unsaturated logic is faster than the saturated logic. The emitter-coupled logic is an example of unsaturated logic. The most commonly used saturated and unsaturated logic circuits are TTL and ECL respectively. For illustration, the TTL NAND gate and ECL NOR gate are considered here.

4.2a Transistor-Transistor Logic Gate

A single multiemitter transistor replaces input diodes and the series diode of DTL [2],[4]. Each emitter-base diode serves

as an input, and the base collector diode functions as the series diode. The multiemitter transistor can be economically fabricated in monolithic form. A single isolated collector region is diffused, a single base region is diffused in the collector region, and several emitter regions are diffused as separate areas into the base region. An output stage using an active pull-up is added, which results in faster switching speed and higher fanout capability. It has got typical delay-time of 10 nano-seconds as against to 50 nano-seconds and 25 nano-seconds of RTL and DTL respectively. Similarly typical fanout of 10 as against to 5 and 8 of RTL and DTL respectively. Power dissipation is 10 milli-watts.

The output active pull-up circuit is having a collector resistance of value 130 Ohms. But such a small valued resistance cannot be fabricated in the selected technology. So it has been replaced by three parallel resistances of equal value.

Figure 4.2 is the circuit schematic of the transistor-transistor logic gate. In the schematic the terminal numbers of the components and the number of terminals which goes to outside world are indicated. The emitters of the multiemitter transistor should be numbered by consecutive numbers. The diode is considered as a three terminal device, the transistor, with third terminal being open circuited.

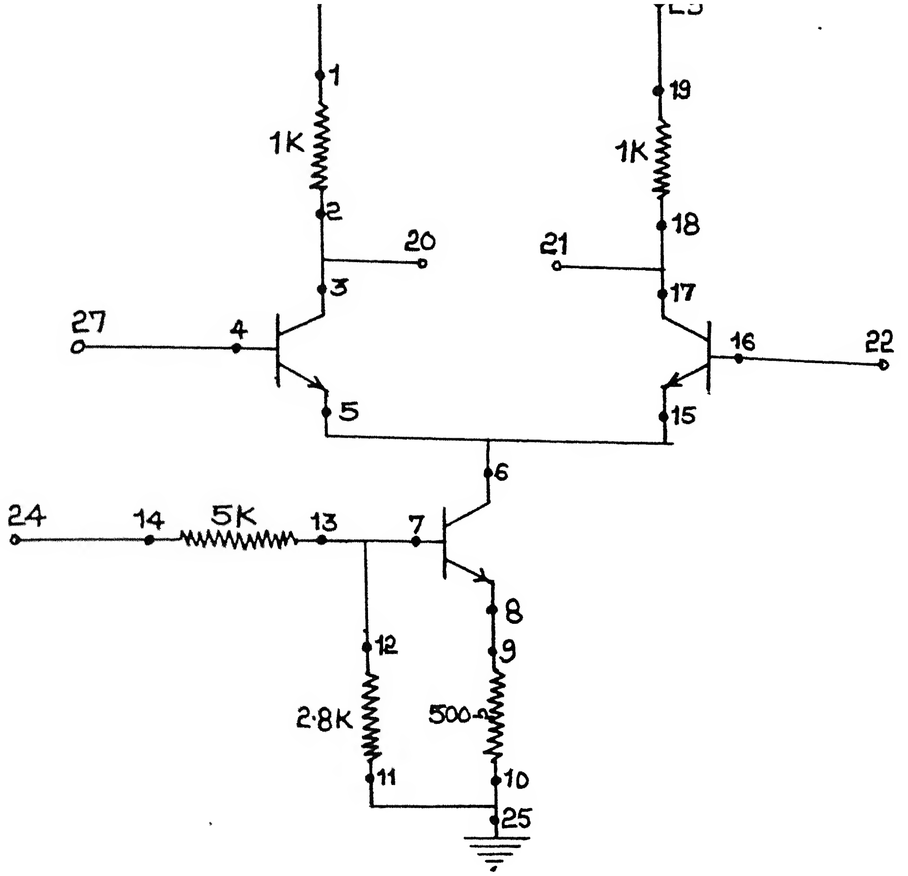


Fig. 4.1 Circuit schematic of differential amplifier of the type CA3028 Differential amplifier

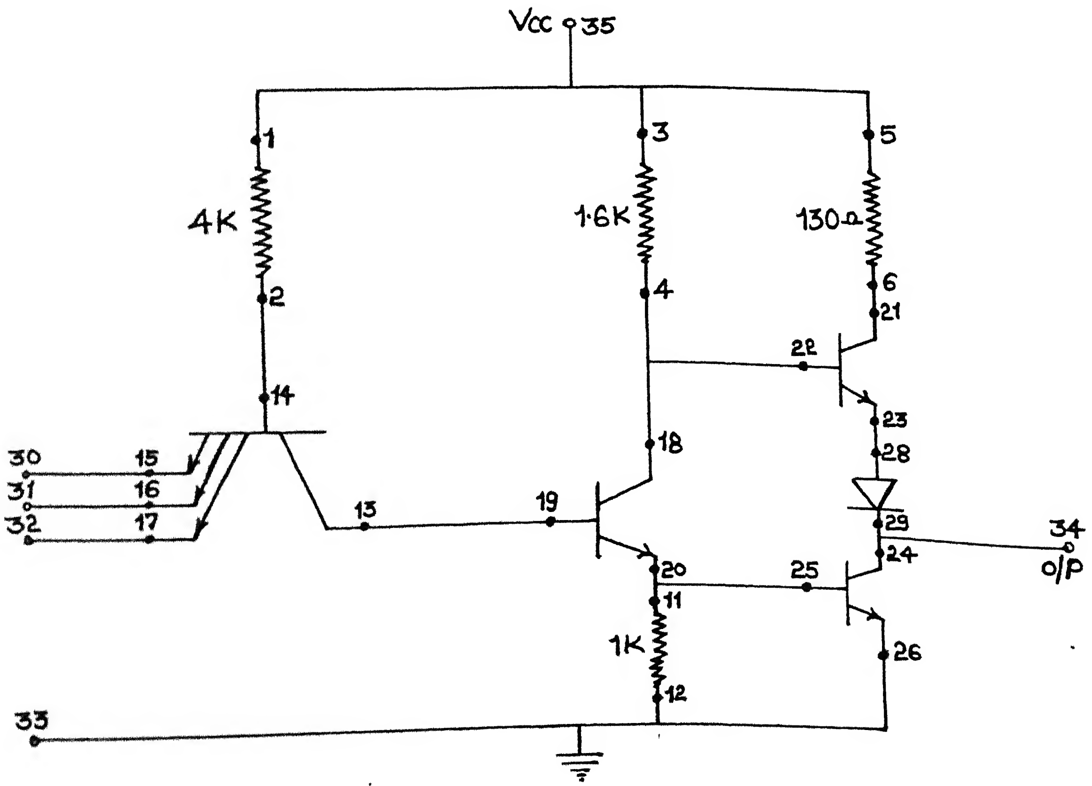


Fig. 4.2 Circuit schematic of transistor-transistor logic gate

4.2b Emitter-Coupled Logic Gate

Emitter coupling prevents the transistors from going into saturation. This results in very fast switching speed, typically only a few nanoseconds. Power dissipation is relatively high, typically 50 milliwatts. It has got high fanout upto 25. ECL is presently used in only the largest computer where many disadvantages can be suffered for the sake of high speed [2],[4].

Figure 4.3 is the circuit schematic of the emitter-coupled logic gate with OR and NOR outputs. In the schematic the terminal numbers of the components and the numbers of terminals which go to outside world are indicated. In the layout diagram 300 Ohms resistance has been replaced by two equal parallel resistances. Similarly it has been done for 270 Ohms resistance.

The input data file along with the layout diagrams drawn by computer for the circuits considered in this chapter are given in Chapter 6. The layout drawings are scaled to 200 times the final size.

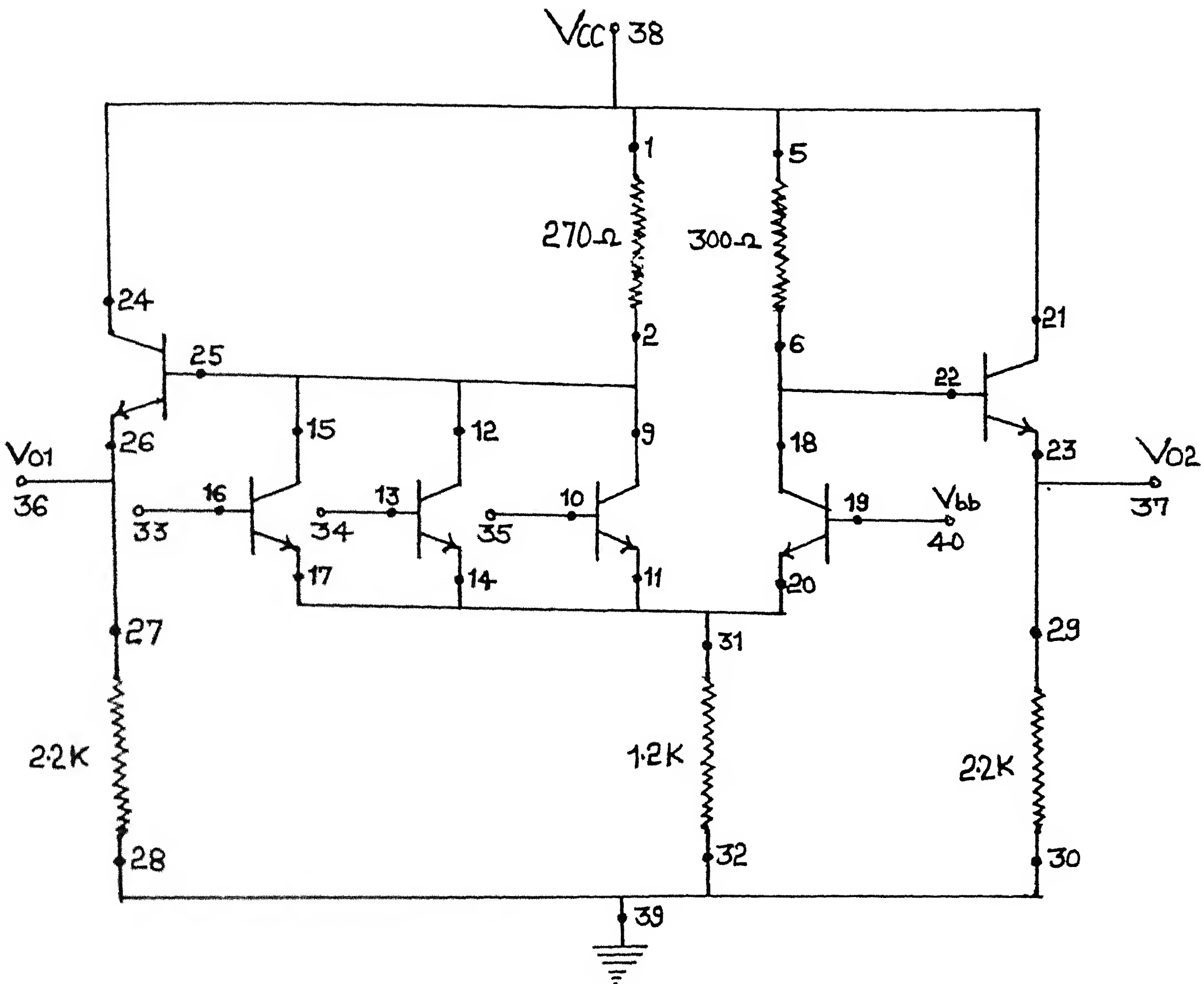


Fig. 4.3 Circuit schematic of emitter-coupled logic gate

CHAPTER 5

SEMI-CUSTOM DESIGN APPROACH TO LSI SYSTEM

When a full-custom design approach is too time-consuming, or not justifiable financially due to low production volume, and when off-the-self packages are too bulky (and consequently too expensive because of the many additional printed circuit boards and large power supplies, or too slow because of long connections), then the semi-custom design approaches are a compromise solution. Currently, the semi-custom design approaches utilize gate arrays, cell library, and others including programmable logic arrays (PLAs).

5.1 GATE ARRAYS

A gate array is an LSI chip on which gatesare placed in matrix from without connections. But strictly speaking, each cell consists of unconnected gates, and are arranged in matrix form and each cell can realise one of certain types of functional modules by connecting these gates. Then by connecting these functional modules, systems can be realised. Only 2 or 3 masks for connections and contacts have to be custom-made, instead of all the masks as for full-custom design. Also because only the connection layout, along with the placement of gates, needs to be considered, computer aided design can be effectively used; greatly reducing the layout time. Thus the design with gate

arrays is inexpensive and quick, compared with the full-custom design. Gate arrays of ECL and TTL have been extensively used in mainframes and minicomputers. Gate arrays are also called master slices or uncommitted logic arrays [5].

5.2 TTL GATE ARRAYS

As for illustration by making use of four NAND gates of the TTL gate array the Exclusive-OR function can be realised. Custom-made metallisation between such NAND gates form the functional module. The logical definition of Exclusive-OR function is

$$F = \overline{A}B + A\overline{B}$$

This boolean expression can be realised by making use of 4 NAND gates as shown in Figure 5.1.

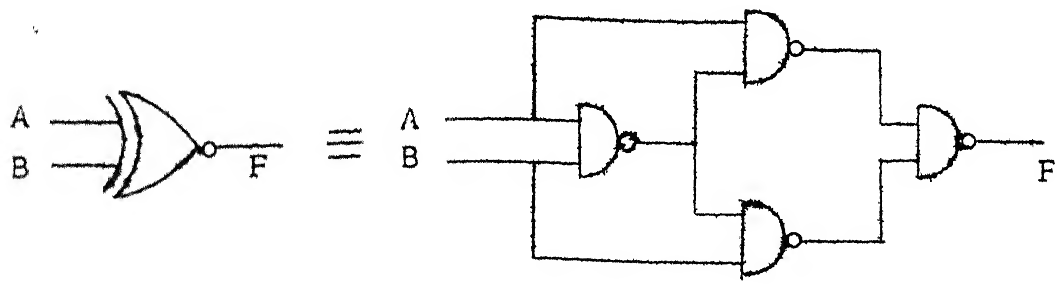


Figure 5.1 Exclusive-OR Function Using 4 NAND Gates

5.3 ECL GATE ARRAYS

Similarly, for illustration, by making use of four NOR gates of the ECL gate array the Exclusive-OR function can be realised. Custom-made metallisation between such NOR gates

form the functional module. The boolean expression for Exclusive-OR function is

$$F = (\overline{A}+\overline{B})\cdot(A+B)$$

$$= \overline{A}B + \overline{B}A$$

Once again this boolean expression can be realised by making use of 4 NOR gates as shown in Figure 5.2.

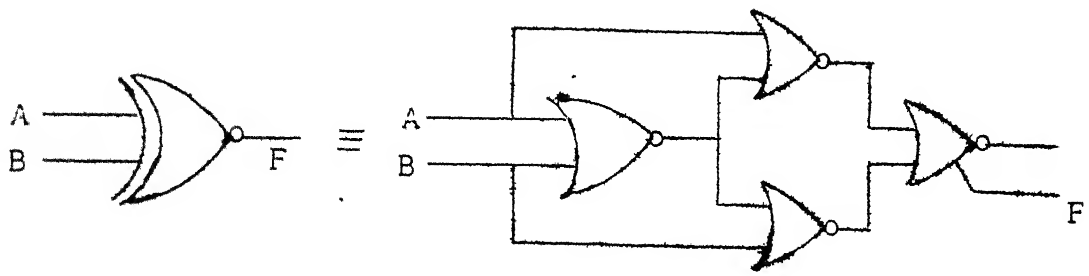


Figure 5.2 Exclusive-OR Function Using 4 NOR Gates

The input data file alongwith the layout diagrams of TTL gate array and ECL gate array drawn by computer are given in the next chapter. The layout drawings are scaled to 100 times the final size.

Due to the limitation of graphic terminal the gate array is limited to the size 2x2. But the software is able to give the layout diagram for nxn gate array , 'n' can take any value but limited by the substrate size. By making use of an exclusive-OR module alongwith the two more gates of the NAND gate array, one can realise a half adder. The realisation of half adder using NAND gates is shown in Figure 5.3. By making use of two half adders one can realise a full adder, which is

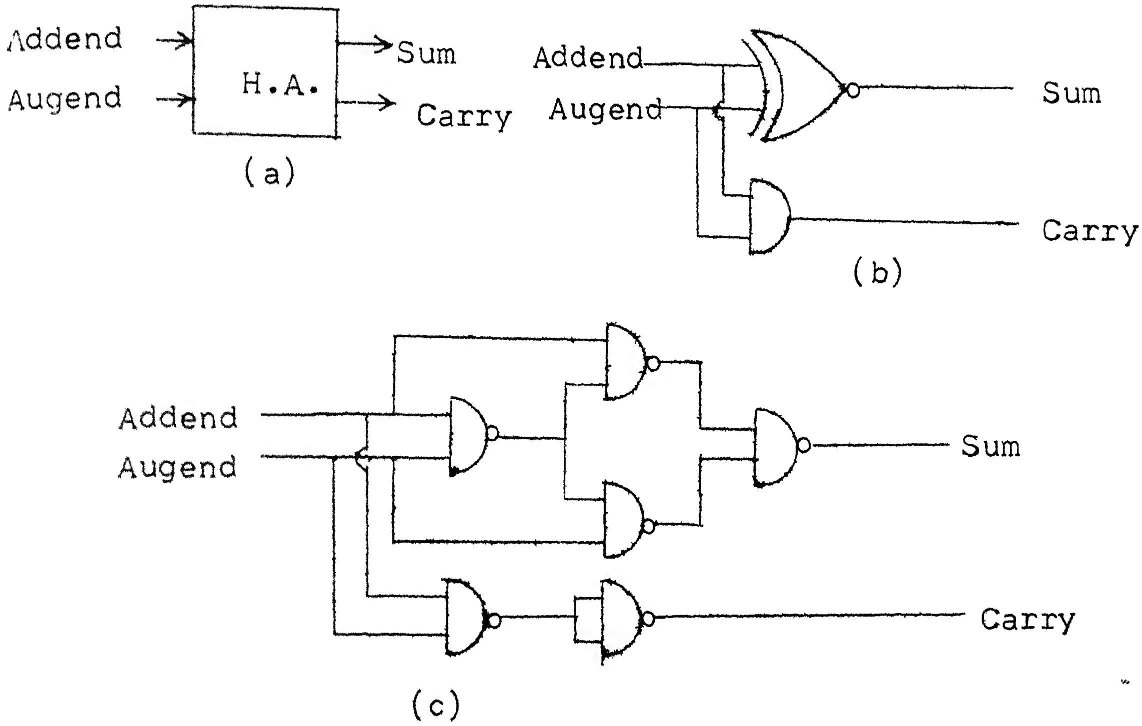


Figure 5.3 Half adder. (a) Block diagram
(b) Symbolic diagram (c) TTL realisation

shown in Figure 5.4. The OR operation can be realised by NAND gates of the array.

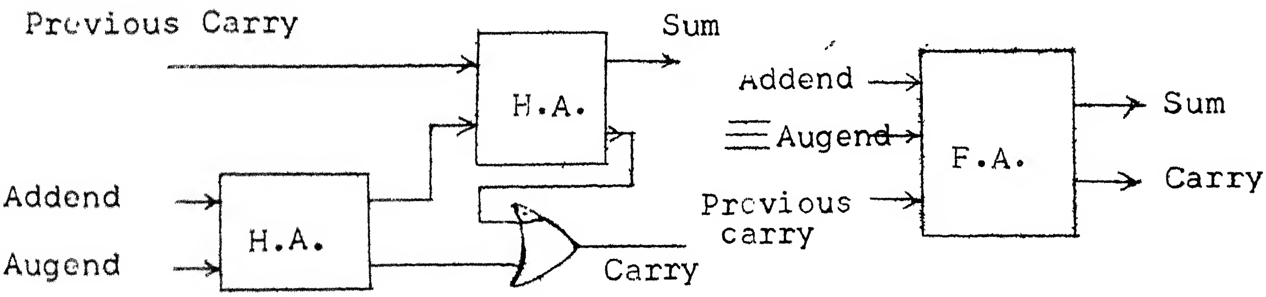


Figure 5.4 Full adder

In this fashion, using gate arrays, one can realise complex combinatorial systems.

By connecting four gates of an array one can get a J-K flip flop. From two such J-K flip flops one can realise a J-K master slave flip flop. Many J-K master slave flip flops of gate array can be used to realise complex functions like counters, shift registers, memories, etc. using additional gates of the array. In this way, by making use of gate array, one can get the sequential circuits, as well.

CHAPTER 6

RESULTS AND CONCLUSIONS

The computer drawn layout diagrams of circuits considered in Chapter 4, TTL, and ECL gate arrays are given in this chapter.

6.1 MASK LAYOUT DIAGRAMS OF DIFFERENTIAL AMPLIFIER

The Table 6.1 is the 'input' datafile for the differential amplifier (of the type CA3028 differential amplifier) explaining about the circuit shown in Figure 4.1. Figures 6.1 through 6.6 are indicating isolation diffusion, base diffusion, emitter diffusion, oxide cut for contacts, metallisation and master diagram respectively. Figure 6.7 shows an array of differential amplifiers. This array size is limited by the size of the substrate. The 2x2 array size has been considered because of the plotter limitation. This figure shows the layout of the quadruple differential amplifier similar to 7400 NAND gates in SN 74/54 series.

6.2 MASK LAYOUT DIAGRAMS OF TRANSISTOR-TRANSISTOR LOGIC GATE

The Table 6.2 is the 'input' datafile for the TTL gate describing about the circuit shown in Figure 4.2. Figures 6.8 through 6.13 indicate isolation diffusion, base diffusion,

emitter diffusion, oxide cut for contacts,metallisation and master diagram respectively.

6.3 MASK LAYOUT DIAGRAMS OF EMITTER-COUPLED LOGIC GATE

The Tables 6.3 and 6.4 are the 'input' and 'inpu22' data-files for the ECL gate describing about the circuit shown in Figure 4.3. Figures 6.14 through 6.19 are indicating isolation diffusion, base diffusion, emitter diffusion, oxide cut for contacts, metallisation and master diagram respectively.

6.4 LAYOUT DIAGRAM OF TTL GATE ARRAY

Table 6.2 is the 'input' datafile of the TTL gate array. Figure 6.20 is the master diagram of the 2x2 TTL gate array.

6.5 LAYOUT DIAGRAM OF ECL GATE ARRAY

Tables 6.3 and 6.4 are the 'input' and 'inpu22' datafiles of ECL gate array. Figure 6.21 is the master diagram of the 2x2 ECL gate array.

Because of the limitation of the plotter size master diagram is given only for 2x2 array. The plots which are shown in this chapter are of limited accuracy according to the plotter's design. Manufacturers of integrated circuits are actively pursuing various methods of obtaining suitable final artwork using numerically controlled artwork generating equipments. Automated equipment using light heads to expose film or sensitized glass are used by IC manufacturers.

Table 6.1 'input' Datafile of Differential Amplifier.

```
10.,10.,2.,10.,200.0
8
1,1,2,-1,1000.,0,1,260.,285.0
1,19,18,-1,1000.,0,1,390.,285.0
1,14,13,-1,5000.,0,1,290.,495.0
1,12,11,-1,2800.,1,1,175.,370.0
1,10,9,-1,500.,1,1,225.,380.0
2,3,4,5,1.,0,-1,225.,190.0
2,17,16,15,1.,0,1,385.,190.0
2,6,7,8,1.,0,-1,335.,370.0
16
3,2
18,17
5,6
6,15
13,12
12,7
11,10
8,9
3,20
17,21
16,22
19,23
14,24
11,25
1,26
4,27
8
20,200.,70.0
21,400.,70.0
22,540.,210.0
23,540.,410.0
24,400.,590.0
26,60.,410.0
27,60.,210.0
25,200.,590.0
70
13
130.,260.0
320.,260.0
320.,310.0
260.,310.0
260.,450.0
420.,450.0
420.,310.0
330.,310.0
330.,260.0
450.,260.0
450.,520.0
130.,520.0
130.,260.0
5
120.,130.0
120.,530.0
460.,530.0
460.,130.0
120.,130.0
```

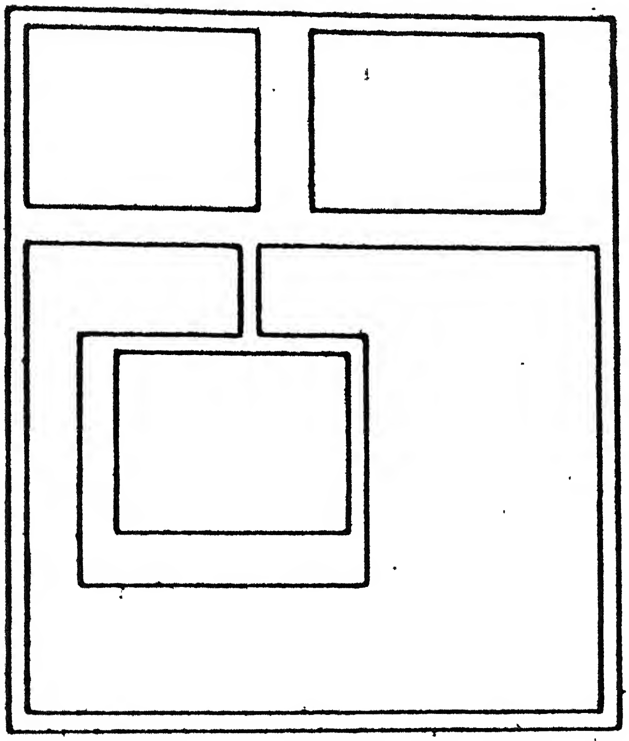


Fig. 6.1 Isolation diffusion

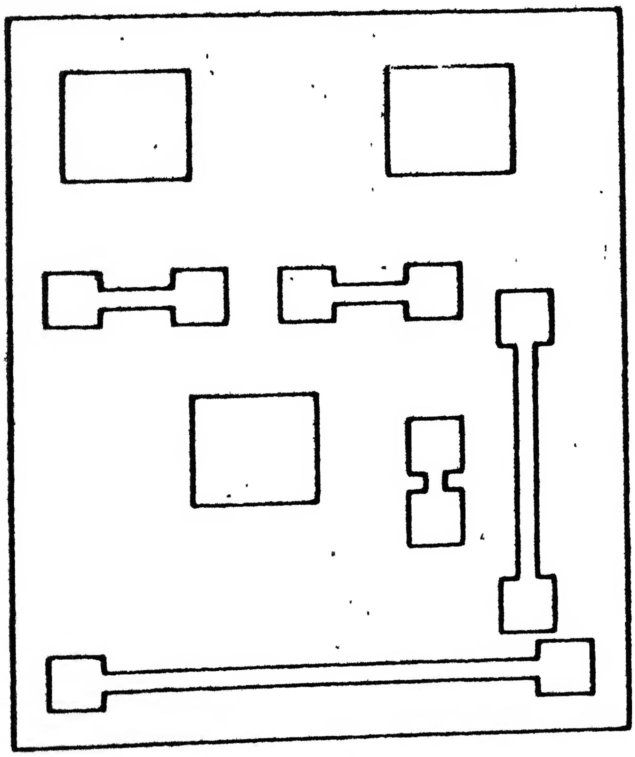


Fig. 6.2 Base diffusion

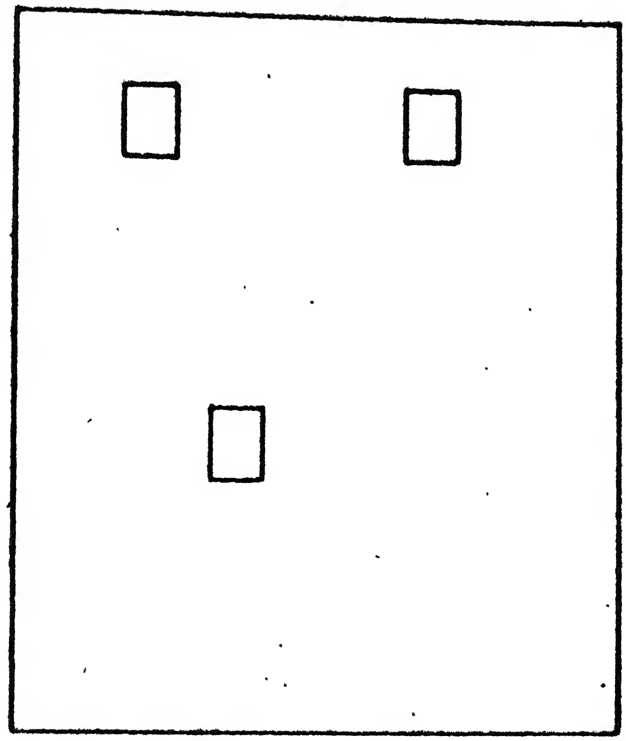


Fig. 6.3 Emitter diffusion

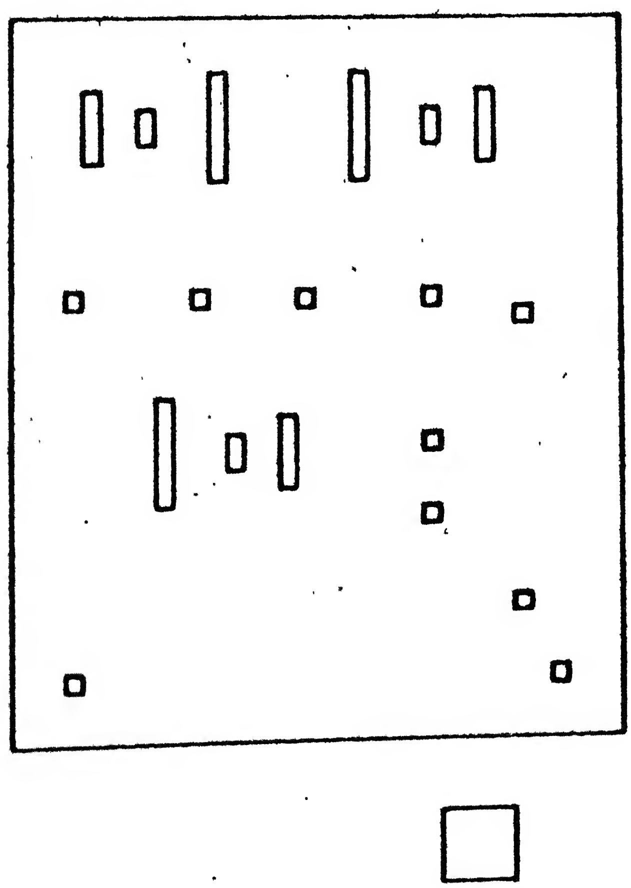


Fig.6.4 Oxide cut for contacts

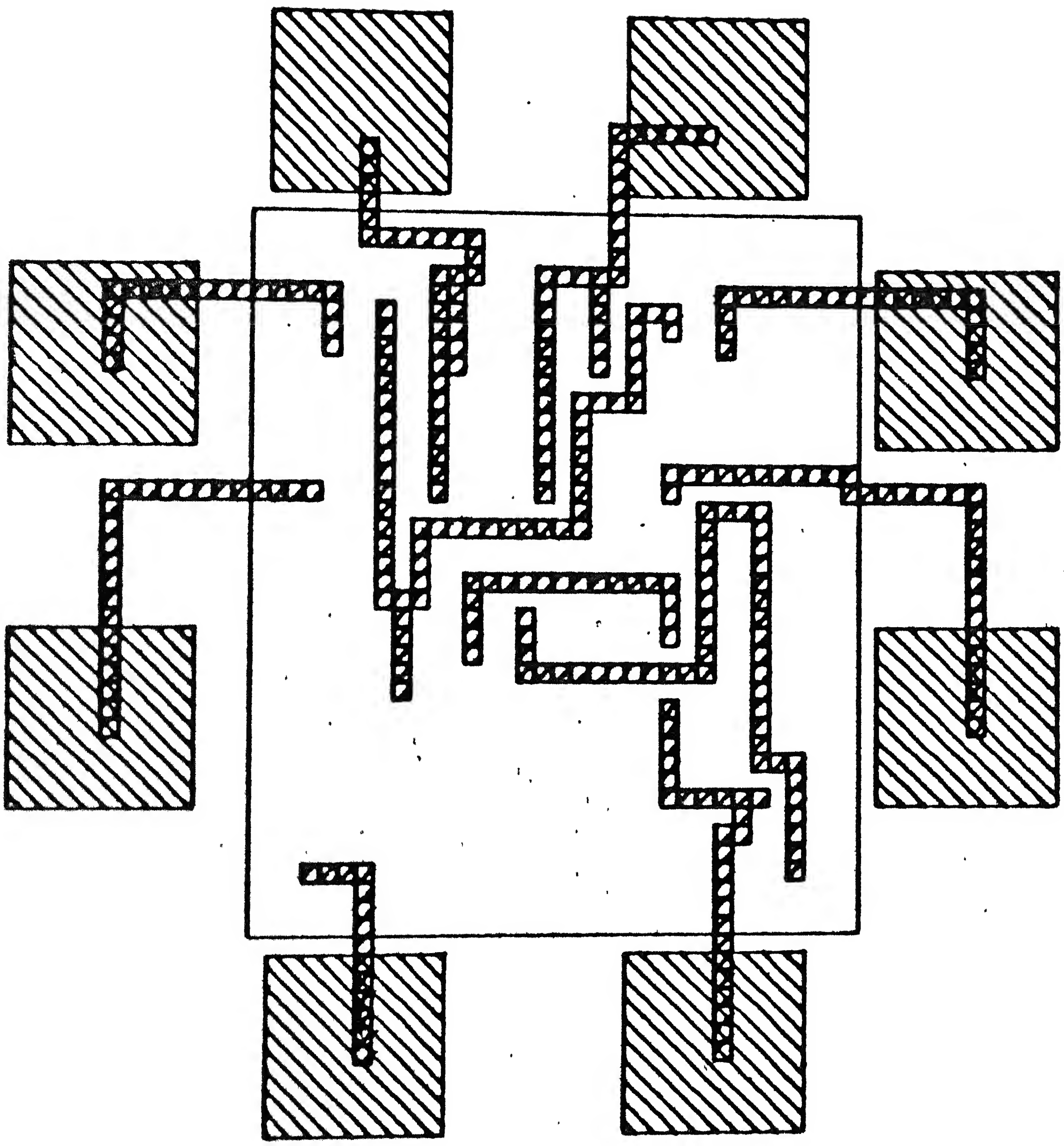


Fig. 6.5 Metallisation

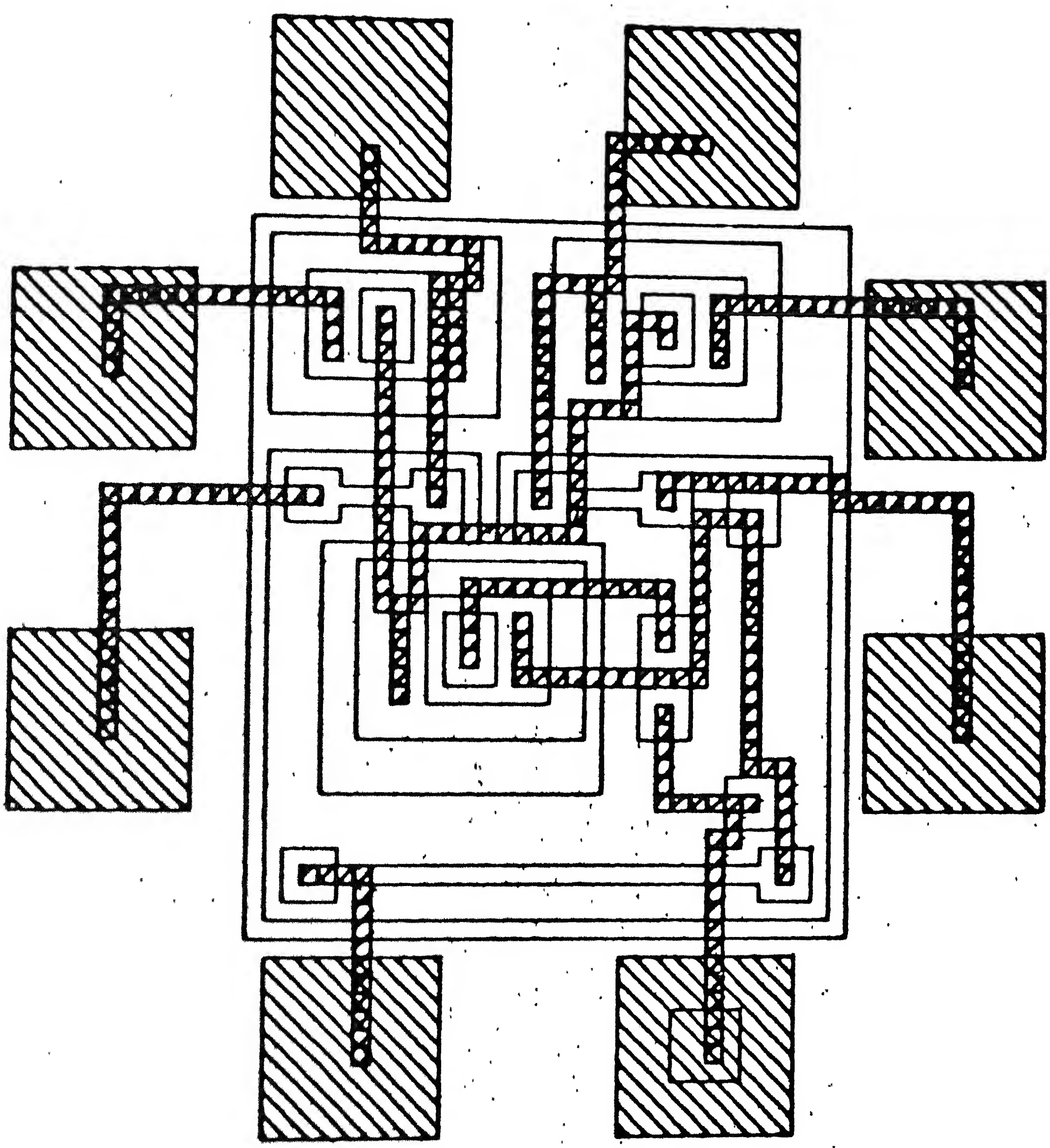


Fig. 6.6 Master diagram

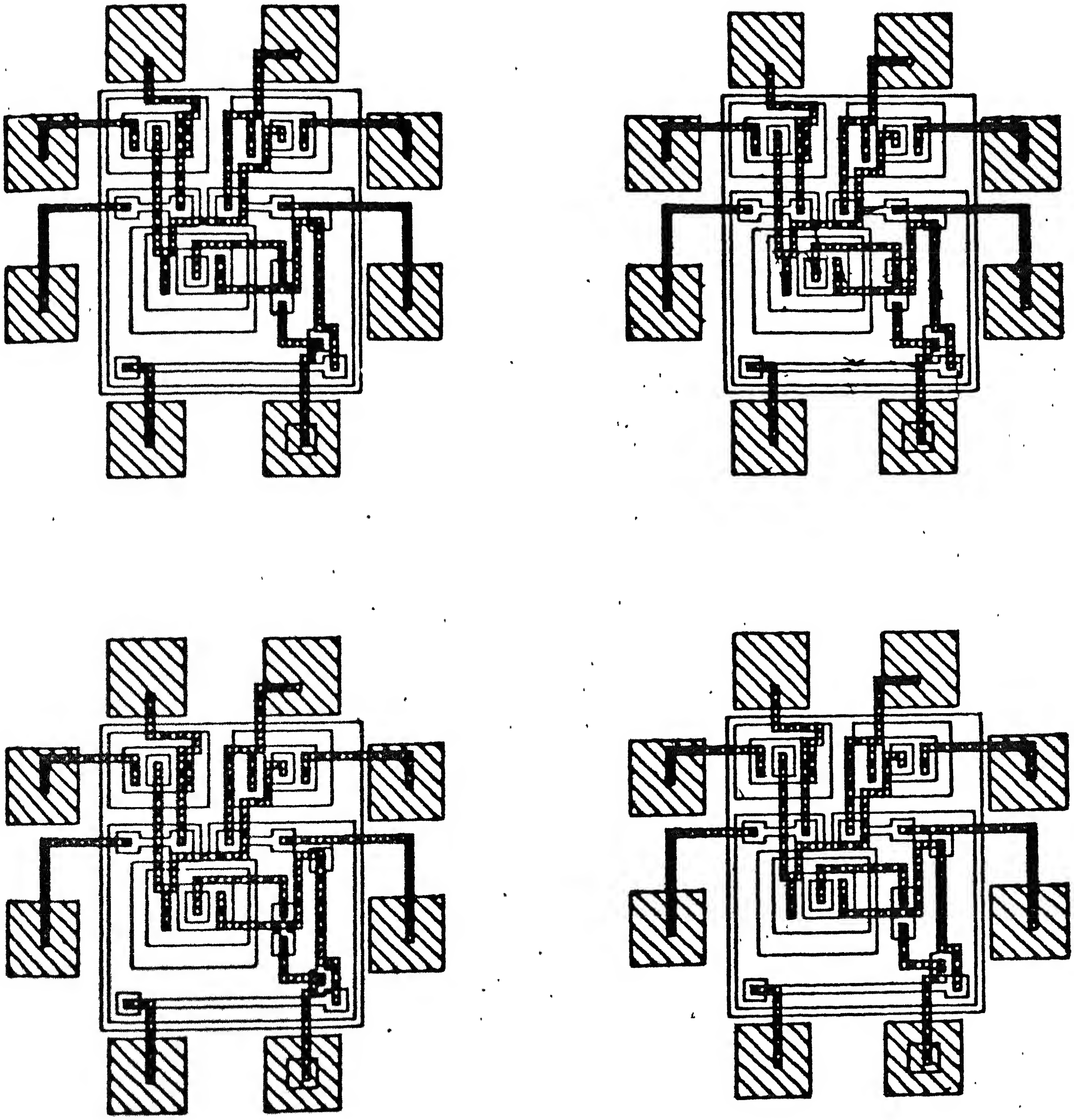


Fig. 6.7 Master diagram of 2x2 differential amplifier array

Table 6.2 'input' Datafile of Transistor - Transistor Logic Gate.

```
10.,10.,2.,10.,200.0
11
1,1,2,-1,4000.,1,1,475.,430.0
1,3,4,-1,1600.,1,1,165.,480.0
1,7,8,-1,100.,1,1,245.,480.0
1,9,10,-1,400.,1,1,285.,480.0
1,5,6,-1,400.,1,1,205.,480.0
1,11,12,-1,1000.,0,1,320.,415.0
4,13,14,15,3.,0,1,405.,200.0
2,18,19,20,1.,0,1,205.,200.0
2,21,22,23,1.,0,-1,205.,320.0
2,24,25,26,1.,0,-1,355.,320.0
3,27,28,29,1.,0,-1,365.,470.0
22
1,3
2,14
4,22
3,7
8,21
7,9
8,10
5,7
6,8
5,35
18,22
23,28
20,25
13,19
15,30
16,31
17,32
11,25
26,12
26,33
29,24
29,34
6
30,70.,180.0
31,220.,70.0
32,450.,70.0
34,580.,450.0
35,70.,450.0
33,580.,200.0
70
7
140.,390.0
440.,390.0

440.,270.0
510.,270.0
510.,570.0
140.,570.0
140.,390.0
5
130.,130.0
520.,130.0
520.,580.0
130.,580.0
130.,130.0
```

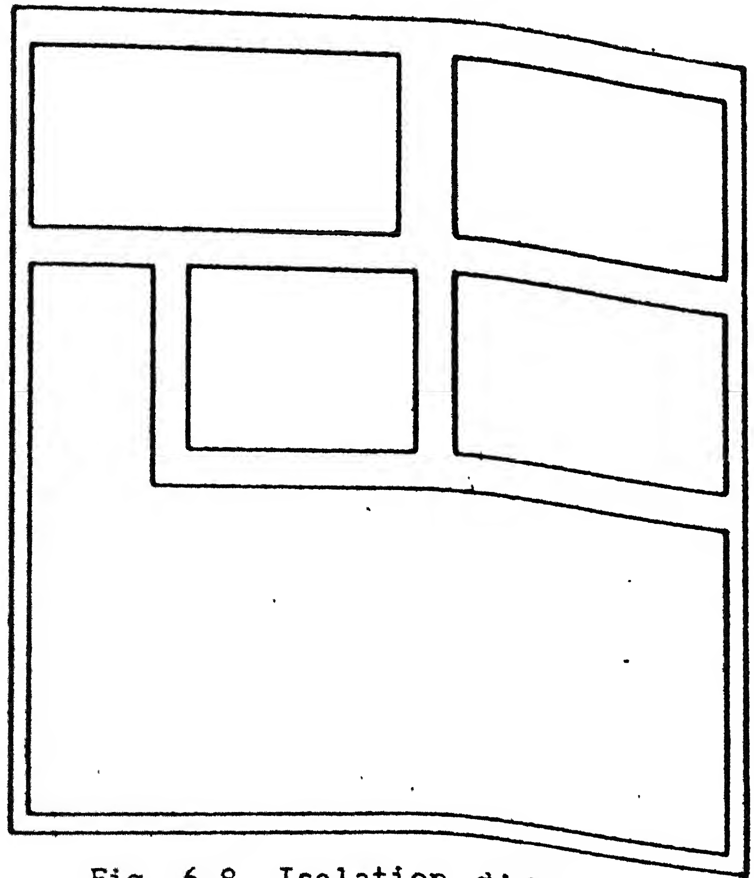


Fig. 6.8 Isolation diffusion

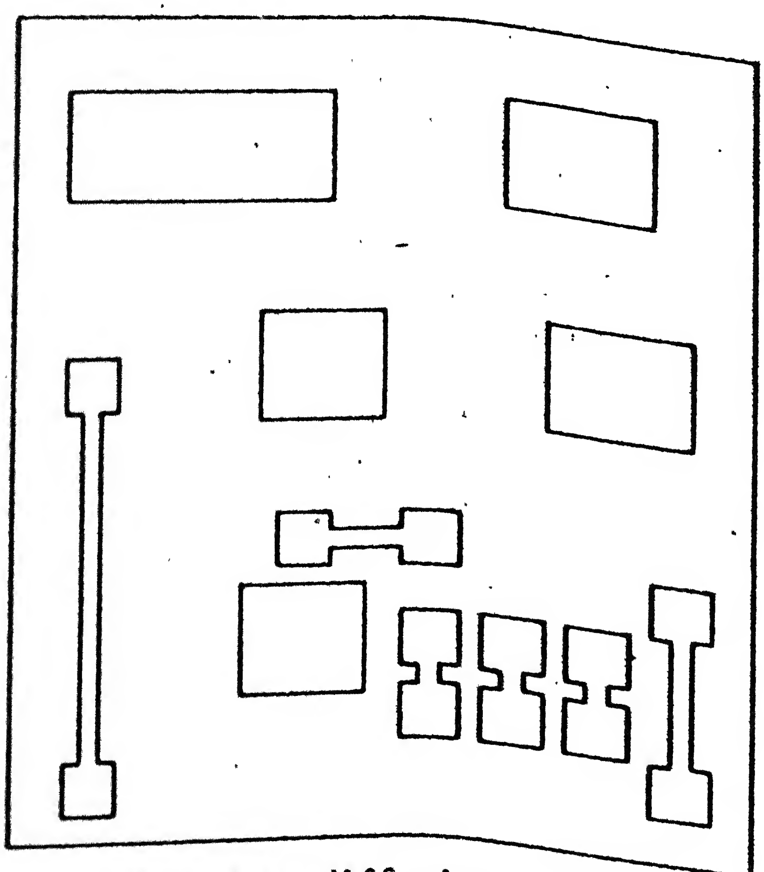


Fig. 6.9 Base diffusion

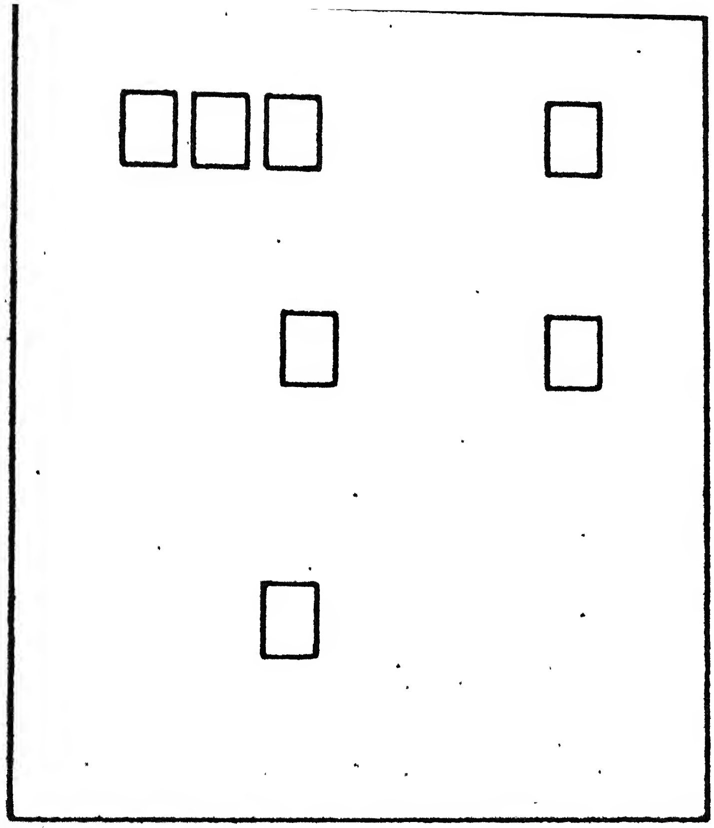


Fig. 6.10 Emitter diffusion

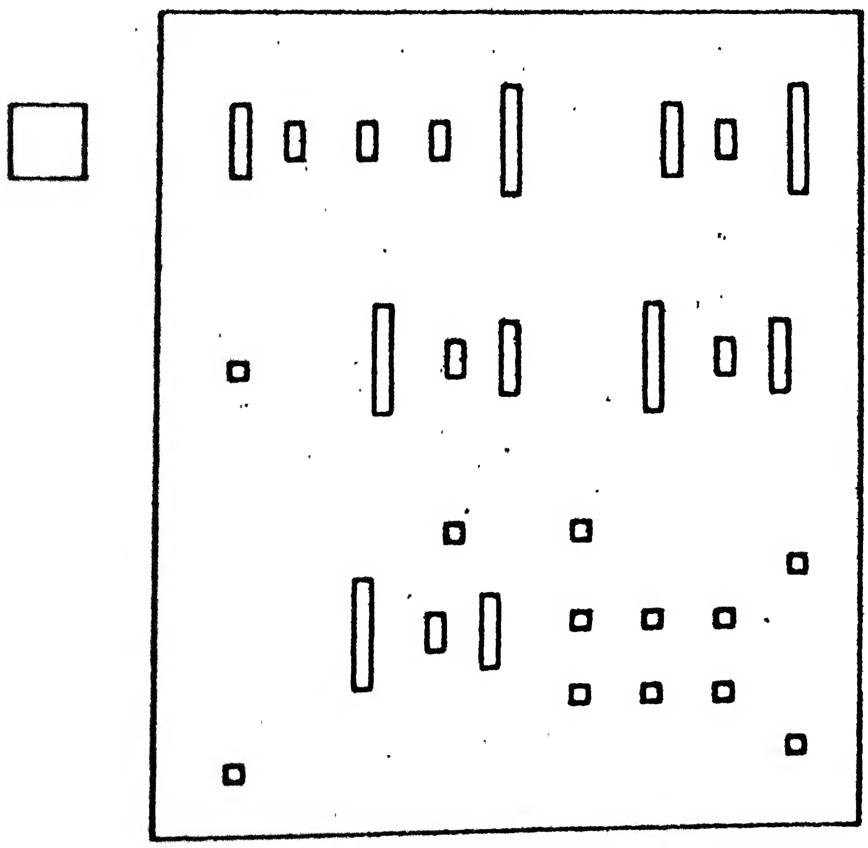


Fig.6.11 Oxide cut for contacts

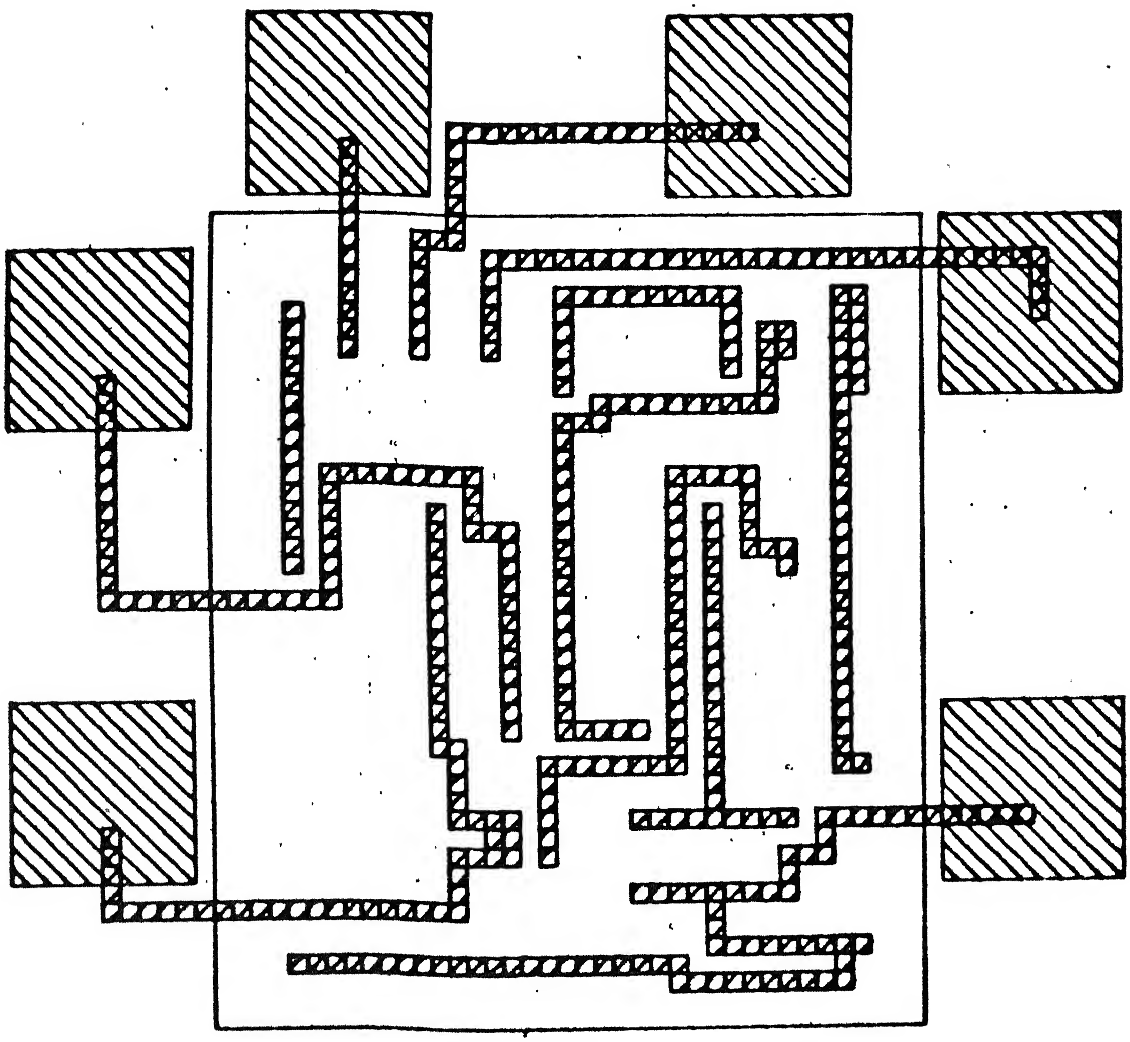


Fig. 6.12 Metallisation

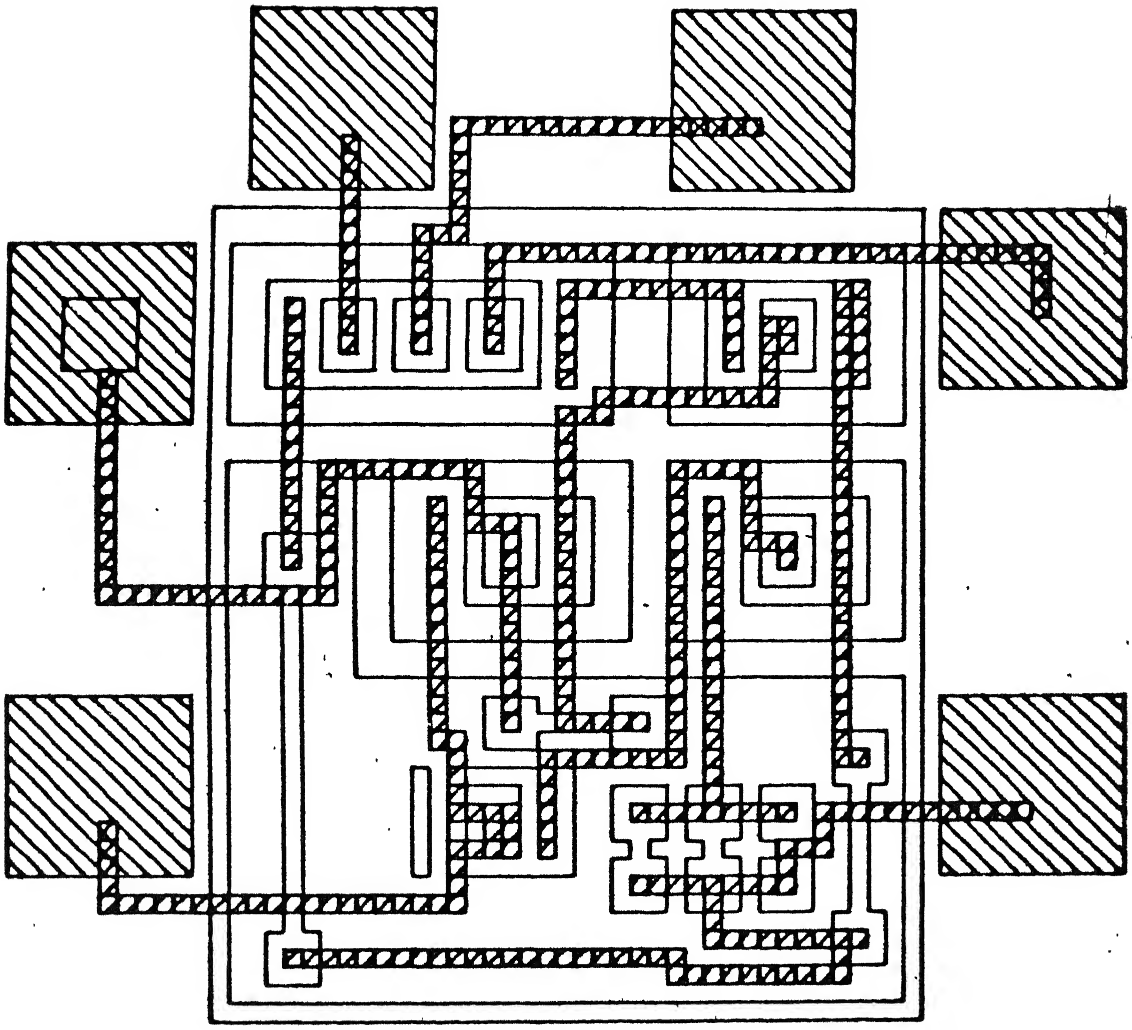


Fig. 6.13 Master diagram

Table 6.3 'input' Datafile of Emitter - Coupled Logic Gate.

```
10.,10.,2.,10.,200.0
13
1,1,2,-1,540.,0,1,330.,415.0
1,3,4,-1,540.,0,1,330.,455.0
1,5,6,-1,600.,0,1,250.,415.0
1,7,8,-1,600.,0,1,250.,455.0
1,27,28,-1,2200.,0,1,330.,495.0
1,29,30,-1,2200.,0,1,330.,535.0
1,31,32,-1,1200.,1,1,165.,400.0
2,18,19,20,1.,1,-1,70.,325.0
3,15,16,17,1.,1,-1,220.,205.0
3,12,13,14,1.,1,-1,290.,205.0
3,9,10,11,1.,1,-1,360.,205.0
3,21,22,23,1.,0,1,285.,340.0
3,24,25,26,1.,0,-1,385.,340.0
28
1,5
2,25
3,1
4,2
6,18
6,8
7,5
27,26
28,32
29,23
30,28
31,20
15,12
12,9
9,25
21,5
7,38
17,31
17,14
14,11
32,39
6,22
19,40
16,33
13,34
10,35
29,37
27,36
8
33,250.,70.0
34,440.,70.0
35,520.,230.0
36,520.,420.0
37,440.,630.0
38,250.,630.0
40,70.,190.0
39,70.,480.0
70
5
140.,290.0
450.,290.0
450.,560.0
140.,560.0
140.,290.0
9
160.,130.0
460.,130.0
460.,570.0
130.,570.0
130.,400.0
10.,400.0
10.,250.0
160.,250.0
160.,130.0
```

Table 6.4 'inpu22' Datafile of Emitter - Coupled Logic Gate.

```
170.,140.0
410.,140.0
410.,270.0
170.,270.0
170.,140.0
```

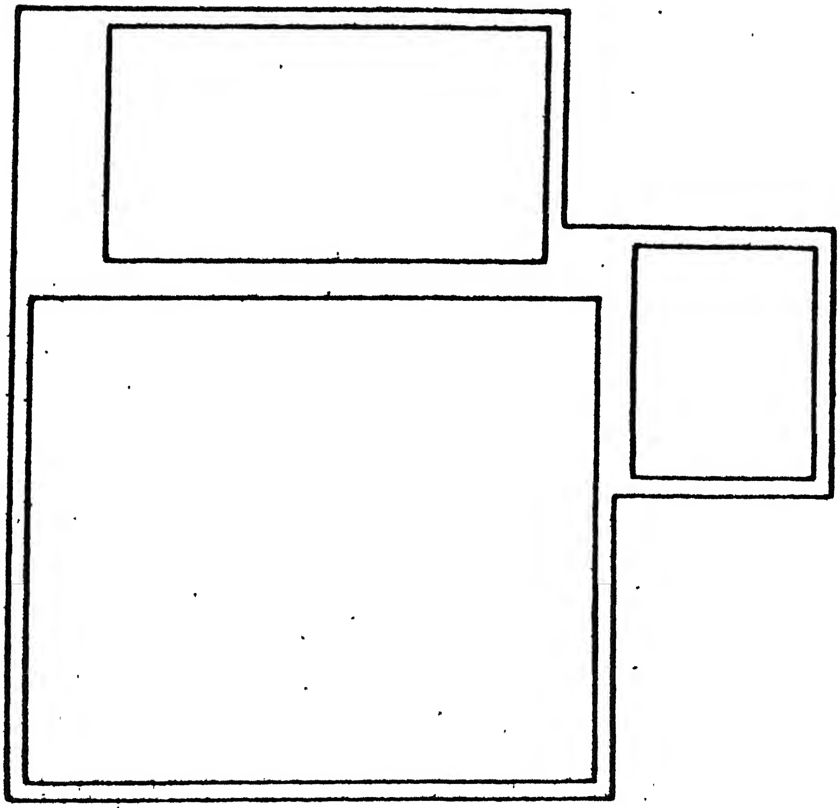


Fig. 6.14 Isolation diffusion

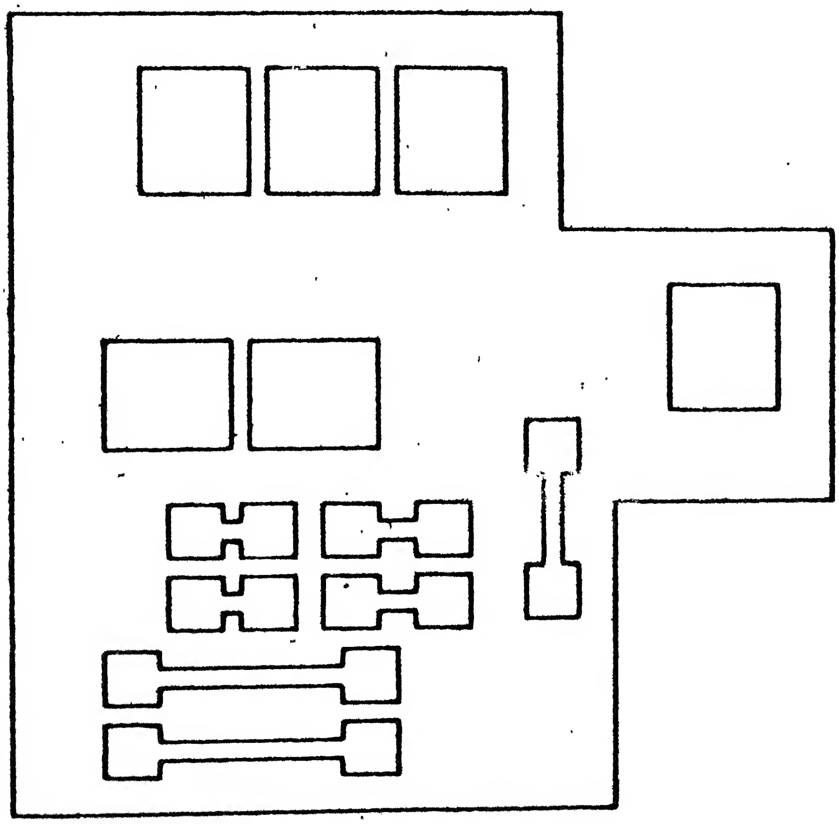


Fig. 6.15 Base diffusion

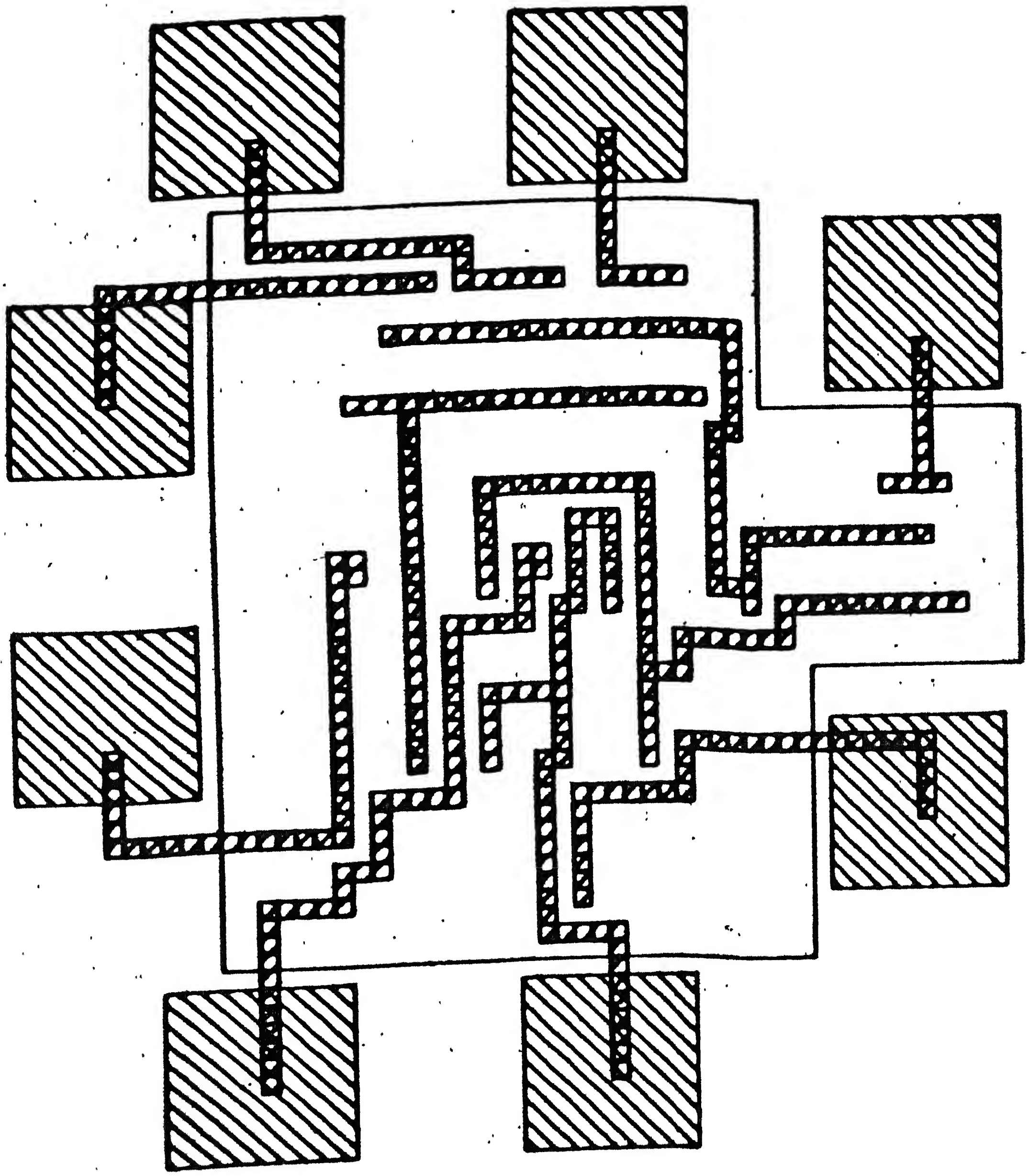


Fig. 6.18 Metallisation

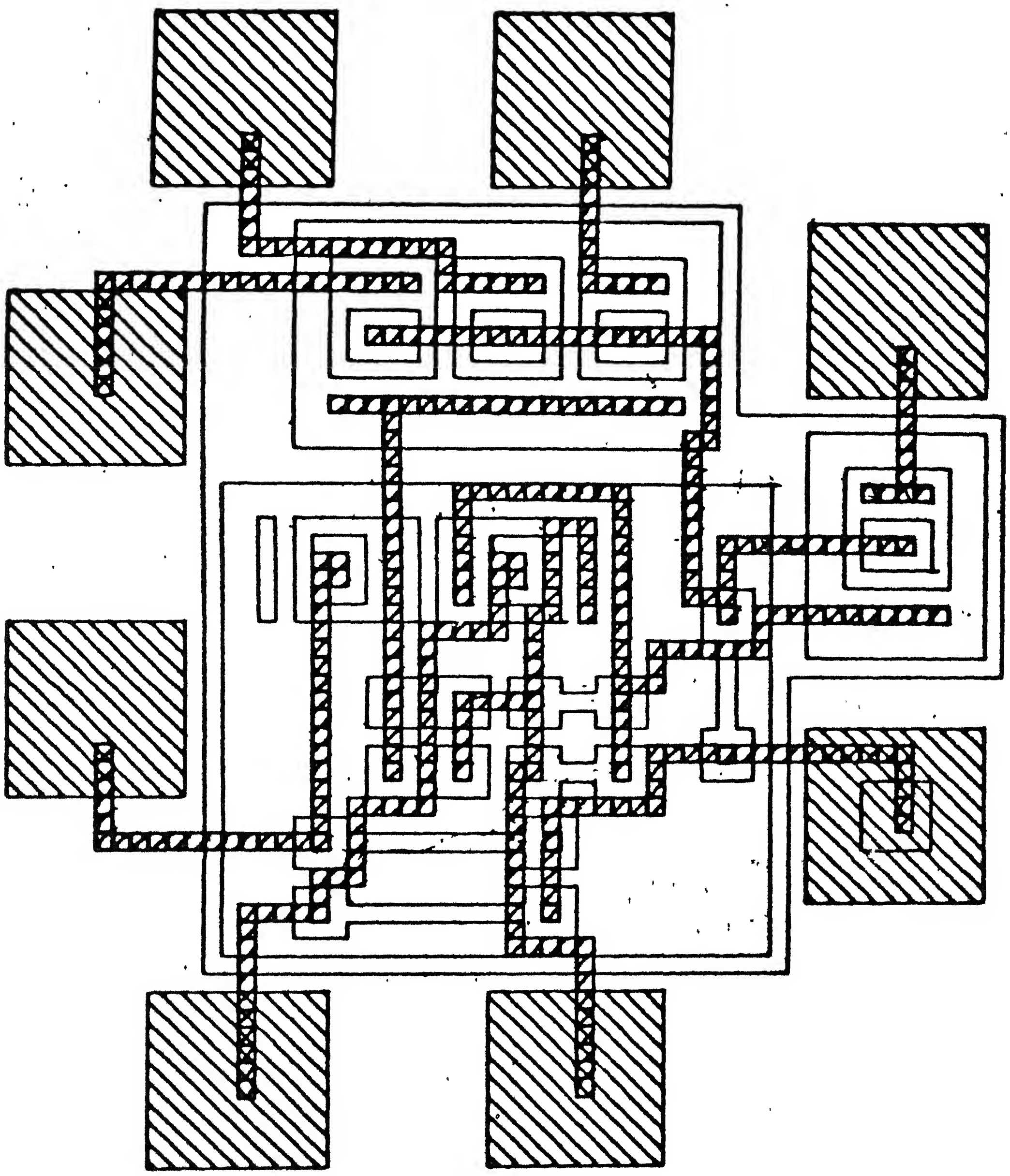


Fig. 6.19 Master diagram

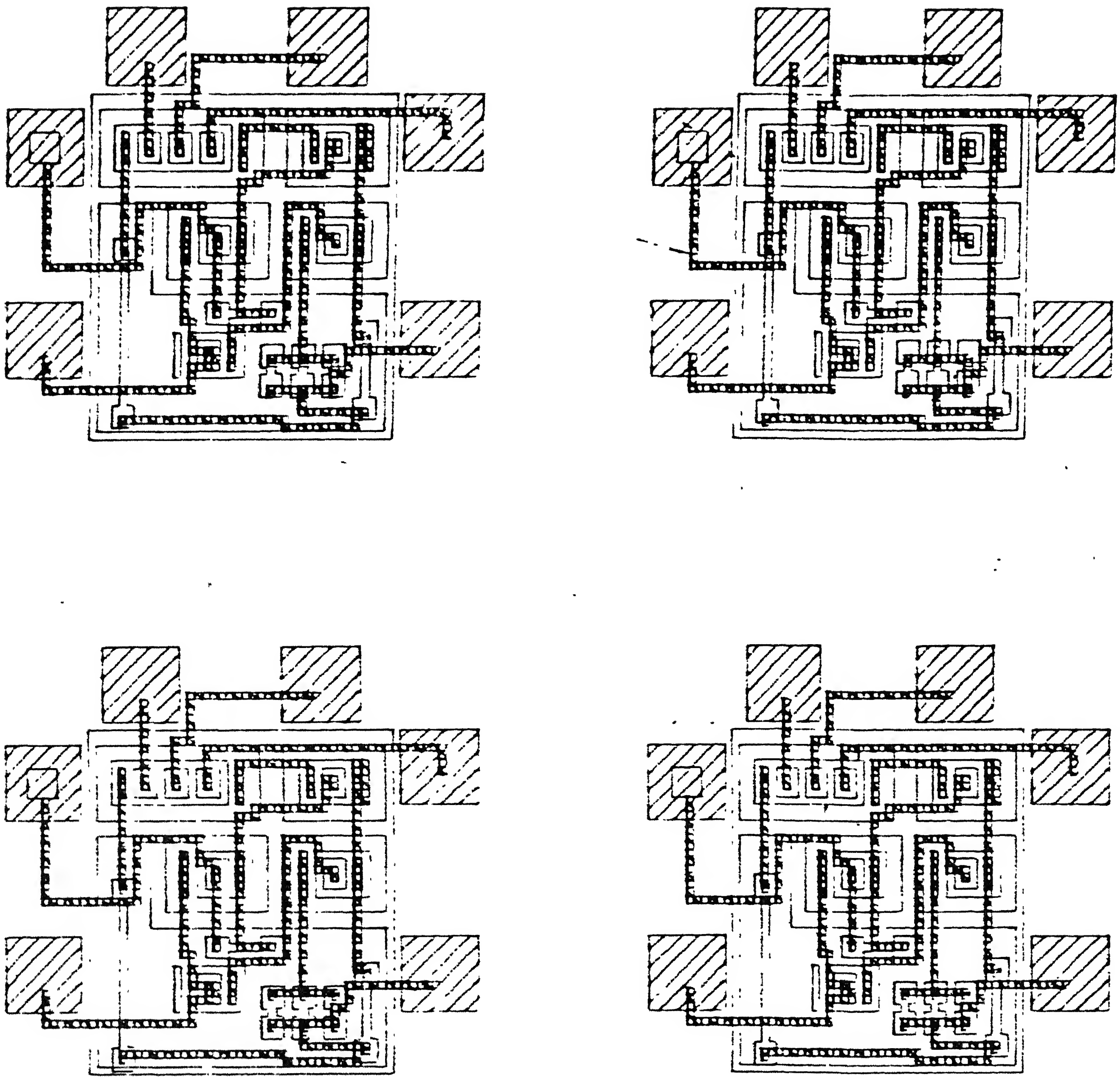


Fig.6.20 Master diagram of 2x2 TTL gate array

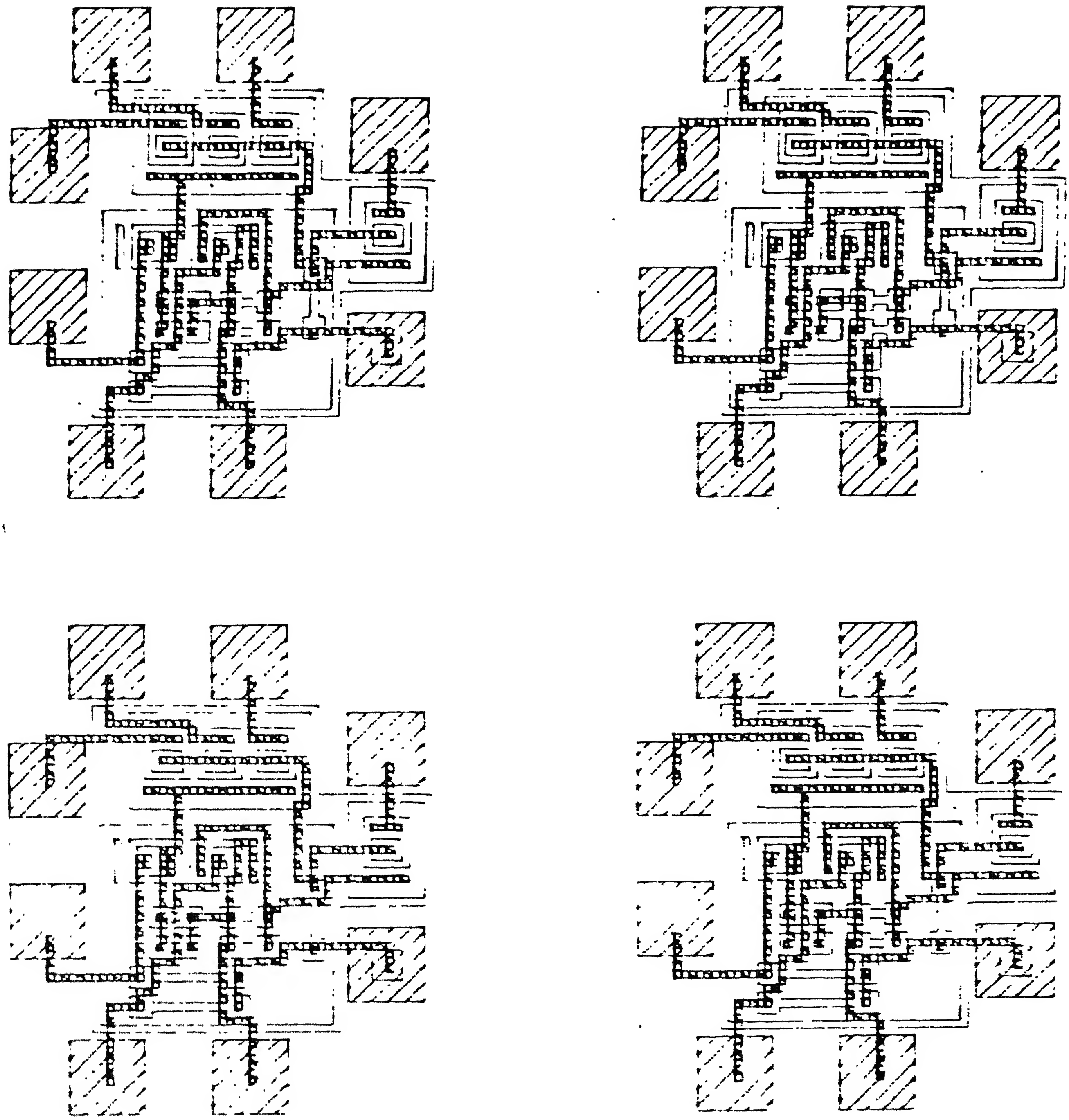


Fig. 6.21 Master diagram of 2x2 ECL gate array

6.6 CONCLUSIONS

An attempt has been made to get the computer aided layout diagram for bipolar integrated circuits. On the similar lines one can attempt to get the layout diagrams for MOS integrated circuits, because the autorouting subroutine is independent of the technology used. This software is capable of handling any sized circuits defined at the circuit level. As an extension to this, one can think of interconnection between the blocks, which are already drawn for the basic circuit. That means one can attempt to get the custom-made masks for interconnection between gates of an array to form functional modules, and between functional modules to form a system. If this is completed, then the package becomes more powerful to give the layout diagrams for MSI, LSI and custom-made circuits. The placement can also be automated though it might require more computation time, and the layout diagrams obtained may not be optimum. One can think of solving for quadratic assignment problem, which is similar to placement problem.

The software is implemented on OMEGA 58000 computer with high resolution graphic terminal. The layout drawing is limited to simple circuits because of the limitation of the graphic terminal. If the graphic terminal is provided with big screen,one can design layout drawings of complex circuits. On the newly installed NORSK DATA-560/CX machine one can design the layout

drawings of more complex circuits with 4109A TEKTRONIX colour graphic terminal. The technology of IC has grown to an extent that a VLSI chip contains over 400000 transistors. Small computers cannot store huge data for such a complex circuits. Thus to design the layout drawings of such a complex circuit super computers like CRAY are needed.

If the autorouter fails to search the destination, instead of changing the position of components reordering of connecting pair of terminals may yield the search.

References

1. Glaser/Subak-Sharpe - 'Integrated circuit Engineering'. Addison-Wesley (1977).

2. D.K. Lynn, C.S. Meyer, D.J. Hamilton - 'Analysis and Design of Integrated Circuits'. McGraw-Hill (1967).

3. Charles F. Wojslaw - 'Integrated Circuits Theory and Applications'. Reston, Raston,Pub. Co. (1978).

4. R.L. Morris and J.R. Miller - 'Designing with TTL Integrated Circuits'. McGraw-Hill (1971).

5. Saburo Muroga - 'VLSI System Design' when and How to Design Very-Large-Scale Integrated Circuits, Wiley Interscience (1982).

6. W.M. Newman, R.F. Sproull - 'Principles of Interactive Computer Graphics'. McGraw-Hill (1973).

7. Irving Dlugatch - 'The Applicability of Computer-Aided Design as a System Engineering Tool'. Proceedings of the IEEE, vol.55, No.11, Nov 1967, pp 1940-1945.

8. Arnold Spitalny, Martin J Goldberg - 'On-Line Graphics Applied to Layout design of Integrated Circuits'. Proceedings of the IEEE, vol.55, No.11, Nov.1967, pp 1982-1988.

9. Arthur Richard Newton - 'Computer Aided Design of VLSI Circuits'. Proceedings of the IEEE, vol.69, No.10, Oct. 1981, pp 1189-1199.

10. Jiri Soukup - 'Circuit Layout'. Proceedings of the IEEE, vol.69, No.10, Oct. 1981, pp 1281-1304.

11. Frank Rubin - 'The Lee Path Connection Algorithm'. IEEE Transactions on Computers, vol.C-23, No.9, Sept. 1974, pp 907-914.

12. Brever, A. Kumar - 'A methodology for custom VLSI Layout'. IEEE Transactions on Circuits and Systems. vol. CAS-30, No.6, June 1983, pp 358.

13. J.H. Hoel - 'Some Variations of Lee's Algorithm'. IEEE Transactions on Computers, Jan. 1976, pp 19-24.

14. C.H . Mays - 'A Brief Survey of Computer Aided Integrated Circuit Layout '. IEEE Transaction on Circuit Theory, vol.CT-18, No.1, Jan. 1971, pp 10-13.

15. S.J. Hong, Ravi Nair - 'Wire-Routing Machines - New Tools for VLSI Physical Design'. Proceedings of the IEEE, vol.71, No.1, Jan. 1983, pp 57-65.

16. PLOT 10 Interactive Graphics Library Manual.

APPENDIX

PLOT 10 Interactive Graphics Library features and some of the routines which have been used in the software development are discussed here.

This library has got seven types of routines.

1. Graphic action routines.
2. Graphic environmental routines.
3. System environmental routines.
4. Text action routines.
5. Text environmental routines.
6. Utility routines.
7. Host file communication routines.

First two types of routines have been made use, so let us see about them slightly more.

Graphic action routines

Graphic action routines are the heart of IGL. These routines generate graphic output to a device. These routines are subdivided into three groups.

Graphic action routines

Graphics output examples :	Graphics input examples :	Miscellaneous Graphics action examples:
MOVE	LOCATE	BELL
DRAW	WHERE	HDCOPY
POLY		HOME
ARC		NEWPAG
ARC3PT		
MPOLAR		
DPOLAR		
PANEL		
MARKER		
FATLIN		

MOVE(PX,PY) - Moves the cursor to the point (PX,PY) without drawing the vector.

DRAW(PX,PY) - Draws the vector from the current cursor position to the point (PX,PY)

POLY(ICNT,XARRAY,YARRAY) - Draws the polygon starting from the first point

ARC(PRAD,PSTARA,PENDA) - Draws an arc with a given radius from the starting angle to the ending angle as indicated, the current cursor position is the centre point for the arc.

ARC3PT(PX2,PY2,PX3,PY3) - Draws an arc starting from the current cursor position through the other two points specified.

MPOLAR(PDIST,PANGLE) - Moves to the specified polar coordinates.

DPOLAR(PDIST,PANGLE) - Draws an arc from the current cursor location to the specified polar coordinates.

PANEL(ICNT,PXARAY,PYARAY)- Displays a panel or an emulated panel on display screen.

MARKER(PX,PY,IMARK) - Displays a symbol at a specified point on the device surface.

FATLIN(ICNT,PX,PY,IN,PD,ITYPE) - Draws wider than normal lines by drawing closely spaced vectors.

LOCATE(IMAXPT,PXARAY,PYARAY,IDAT,IGOT) - Puts the terminal into graphic input (GIN) mode and stores coordinates of points located by the graphic cursor.

WHERE(PX,PY) - Returns the location of the cursor in world space units.

BELL - Causes the device's bell to sound.

HDCOPY - Activates an attached hardcopy unit and copies the screen.

HOME - Moves the cursor to the 'home' position.

NEWPAG - Provides a clean surface for display of output.

Graphic Environmental routines

A graphics environmental routine is one that establishes a condition controlling the display of IGL graphic output. The subdivision of these routines is as under.

Graphic Environment Routines →

- → Transform examples : VIEWT, MODELT, PIVOT
- → Modelling transformation examples: SCALE, ROTATE, TRANSL, MTRAN
- → Viewing transformations examples: WINDOW, VWPORT, TRIDNT
- → Surface display and angle units. examples: MILLIM, INCHES, RASTER, GDUNIT, RADIAN, PIRAD, GRADS, DEGREE
- → Displacement examples : VECREL, VECABS
- → Clipping examples : EDGE, NOCLIP, CLIP
- → Vector environment examples: DASHPT, SKIP, REMOVE, APPEAR, GRAIN, CLRMAP, LINCLR, BKGCLR
- → Master color examples : MRKCLR, FILPAN

VIEWT - Specifies that subsequent scaling, rotation, and translation transforms should occur only at the viewing surface.

MODELT - Specifies that subsequent scaling, rotation, and translation transforms should occur in world space as modelling transforms.

PIVOT(PXINV,PYINV) - Specifies the pivot point for transforms

SCALE(PXSC,PYSC) - Specifies a scale factor applied to the coordinate system

ROTATE(PANGLX,PANGLY)- Indicates rotation angles to be applied to the X- and Y-axes

TRANSL(PXDISP,PYDISP) - Applies specified translation (displacement) to coordinates

MTRAN(PTM,QRELTR) - Applies a given matrix transform

WINDOW(XMIN,XMAX,YMIN,YMAX) - Specifies the portion of the coordinate system to be viewed.

VWPORT(XMIN,XMAX,YMEN,YMAX) - Defines location of output on the display surface.

TRIDNT(QFULL) - Resets the modeling transform, the window / viewport transform, or both to identify initial values.

MILLIM - Specifies that values in subsequent IGL routines which refer to the display surface are interpreted as millimeters.

INCHES - Specifies that values in subsequent routines which refer to the display surface are interpreted as inches.

RASTER- Specifies that values in subsequent routines which refer to the display surface an interpreted as rasters.

GDUNIT - Specifies that values in subsequent routines which refer to the display surface are interpreted as graphics display units. It is the default.

RADIAN - Declares that angles are specified in radians .

PIRAD - Declares that angles are specified in pi-radians .

GRADS - Declares that angles are specified in gradians.

DEGREE - Declares that angles are specified in degrees.

VECREL - Declares that subsequent coordinates are relative to the cursor position.

VECABS - Declares that subsequent coordinates are absolute (i.e., relative to the origin).

EDGE (XMIN,XMAX,YMIN,YMAX) - Defines the clipping window independently of the viewport.

NOCLIP - Terminates clipping.

CLIP - Activates clipping.

DASHPT(IPAT) - Specifies pattern for dashed lines.

SKIP - Makes the next vector invisible .

REMOVE - Specifies selective erase mode.

APPEAR - Returns device from selective erase mode.

GRAIN(PGRAIN) - Specifies the granularity (roundness) of an arc.

CLRMAP (IMAPNO,ITYPE,PCOLOR) - Specifies a colour to be updated or added to the colour index.

LINCLR(ICOLOR) - Specifies the desired line colour.

BKGCLR(ICOLOR) - Specifies the desired background colour.

MRKCLR(ICOLOR) - Sets marker color.

FILPAN(IPATNO,QOUTLN) - Specifies way in which panels are filled.

For the other routines one can refer the PLOT 10 manual given in the reference list.

PROGRAM LISTINGS
****

```
C       *****************************************************************
C
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C       THE PROGRAM FOR DRAWING THE LAYOUT OF A BIPOLAR INTEGRATED CIRCUIT.
C       THIS PROGRAM CALLS ELEVEN SUBROUTINES APART FROM INTERACTIVE GRAPH-
C       IC LIBRARY PLOT 10 ROUTINES.'COLLET' IS THE SUBROUTINE TO DRAW LAY-
C       OUT OF COLLECTOR DIFFUSION OR IT IS ALSO CALLED AS ISOLATION DIFFU-
C       SION.'BASE' IS THE SUBROUTINE TO DRAW LAYOUT OF BASE DIFFUSION.'EM-
C       ITR' IS THE SUBROUTINE TO DRAW LAYOUT OF EMITTER DIFFUSION.'METOPN'
C       IS THE SUBROUTINE TO DRAW LAYOUT OF METALISATION OPENING.'PAD' IS
C       THE SUBROUTINE TO DRAW LAYOUT OF CONNECTING PADS AND ALSO TO INITI-
C       ALISE THE ROUTING MATRIX.'MATXI' IS THE SUBROUTINE TO INITIALISE
C       THE MATRIX OF LEE'S ROUTER FOR AUTOROUTING.'METLN' IS THE SUBROUTI-
C       NE TO DO THE METALISATION OF INTERCONNECTIONS BETWEEN THE COMPONEN-
C       TS OF THE CIRCUIT.INTURN THE 'METLN' WILL CALL ANOTHER SUBROUTINE
C       BY THE NAME 'LEESR'.'LEESR' IS THE SUBROUTINE FOR LEE'S ROUTER.THI-
C       S LEE'S ROUTER INTURN CALLS TWO MORE SUBROUTINES 'TEST1' AND 'RETR'.
C       HERE 'RETR' STANDS FOR RETRACE.'WAIT' IS THE SUBROUTINE FOR HALTING
C       THE EXECUTION OF THE PROGRAME TO FEED THE COMMAND INTERACTIVELY FOR
C       GETTING MASKS.TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
C       LIST1X,LIST1Y,LIST2X,LIST2Y ARE THE INTEGER COMMON ARRAYS OF TYPE
C       OF THE COMPONENT,ORIENTATION OF THE COMPONENT IN THE LAYOUT,
C       SENS OF THE COMPONENT i.e IS  IT NORMAL OR INVERTED,TERMINAL 1 OF
C       THE COMPONENT,TERMINAL 2 OF THE COMPONENT,TERMINAL 3 OF THE COM-
C       PONENT,CONNECTION INFORMATION OF THE NODES,CONNECTING PAD NUMBERS,
C       TERMINAL ADDRESS IN THE GRID STRUCTURE i.e IN THE ROUTING MATRIX OF
C       THE LEES'S ROUTER,ROUTING MATRIX,COLUMN INFORMATION OF THE CELL
C       TO BE EXPANDED,ROW INFORMATION OF THE CELL TO BE EXPANDED,COLUMN INF-
C       ORMATION OF THE CELL WHICH IS VISITED DURING EXPANSION,AND ROW INFOR-
C       MATION OF CELL WHICH IS VISITED DURING EXPANSION RESPECTIVELY.
C       VAL,POSI,PADPO,TRAND,RESID ARE THE REAL COMMON ARRAYS OF VALUE OF
C       THE COMPONENT,POSITIONAL CO-ORDINATES OF COMPONENT,POSITIONAL CO-OR-
C       DINATES OF PADS,DATA BASE OF TRANSISTOR,AND DATA BASE OF RESISTOR
C       RESPECTIVELY.XJS,M, AND K ARE THE DESIGN PARAMETERS OF RESISTOR.
C       N IS THE NUMBER OF COMPONENTS IN THE CIRCUIT.N1 IS THE NUMBER OF IN-
C       TERCONNECTIONS.N2 IS THE NUMBER OF PADS.N3 IS THE ROUTING MATRIX
C       DIMENSION.N4 IS THE NUMBER OF CO-ORDINATES OF THE RESISTOR ISLAND.
C       N5 IS THE NUMBER OF CO-ORDINATES OF CIRCUIT BOUNDARY.
C       BORDRX,BORDRY,RILNDX,RILNDY,STRAIX,STRAIY ARE THE REAL ARRAYS OF
C       X CO-ORDINATE OF CIRCUIT BOUNDARY,Y CO-ORDINATE OF CIRCUIT BOUNDARY,
C       X CO-ORDINATE OF RESISTOR ISLAND,Y CO-ORDINATE OF RESISTOR ISLAND,
C       X CO-ORDINATE OF AN ISLAND WHICH CONTAINS MORE THAN ONE TRANSISTOR.
C       Y CO-ORDINATE OF AN ISLAND WHICH CONTAINS MORE THAN ONE TRANSISTOR
C       RESPECTIVELY.
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1  TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2  LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
        REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
        COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1  LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
        DIMENSION BORDRX(20),BORDRY(20),RILNDX(20),RILNDY(20),
     1  STRAIX(5),STRAIY(5)
        REAL XJS,M,K,W,RS
        OPEN(UNIT=10,FILE='input',RECL=80,STATUS='OLD',ACCESS='SEQUENT
     1  IAL',FORM='FORMATTED')
        READ(10,10)XJS,M,K,W,RS
10      FORMAT(5F8.1)
        READ(10,12)N
12      FORMAT(I5)
        READ(10,14)(TYPE(I),TERM1(I),TERM2(I),TERM3(I),VAL(I),ORINT(I),
     1  SENS(I),POSI(I,1),POSI(I,2),I=1,N)
14      FORMAT(4I5,F8.1,2I3,2F8.1)
        READ(10,12)N1
        READ(10,16)(CONN(I,1),CONN(I,2),I=1,N1)
16      FORMAT(2I5)
        READ(10,12)N2
        READ(10,18)(PADN(I),PADPO(I,1),PADPO(I,2),I=1,N2)
18      FORMAT(I5,2F8.1)
        READ(10,12)N3
        READ(10,12)N4
```

```
      READ(10,20)(RILNDX(I),RILNDY(I),I=1,N4)
20    FORMAT(2F8.1)
      READ(10,12)N5
      READ(10,22)(BORDRX(I),BORDRY(I),I=1,N5)
22    FORMAT(2F8.1)
      CLOSE(UNIT=10,STATUS='OLD')
      DO 30 I=1,N3
      DO 30 J=1,N3
30    MATX(I,J)=0
      TRAND(1,1)=-1*(5.5*M+XJS)
      TRAND(1,2)=-1*(3.*M+K*M/2.+XJS)
      TRAND(2,1)=9.*M+2.*(XJS+M)
      TRAND(2,2)=0
      TRAND(3,1)=0.0
      TRAND(3,2)=(K+4.)*M+2.*(XJS+M)
      TRAND(4,1)=-1*(9.*M+2.*(XJS+M))
      TRAND(4,2)=0.0
      TRAND(5,1)=0.0
      TRAND(5,2)=-1*TRAND(3,2)
      TRAND(6,1)=-2.5*M
      TRAND(6,2)=-1*(2.+K/2.)*M
      TRAND(7,1)=7.*M
      TRAND(7,2)=0.0
      TRAND(8,1)=0.0
      TRAND(8,2)=(K+4.)*M
      TRAND(9,1)=-7.*M
      TRAND(9,2)=0.0
      TRAND(10,1)=0.0
      TRAND(10,2)=-1*TRAND(8,2)
      TRAND(11,1)=-1.5*M
      TRAND(11,2)=-1*(M+K*M/2.)
      TRAND(12,1)=3.*M
      TRAND(12,2)=0.0
      TRAND(13,1)=0.0
      TRAND(13,2)=(K+2.)*M
      TRAND(14,1)=-3.*M
      TRAND(14,2)=0.0
      TRAND(15,1)=0.0
      TRAND(15,2)=-1*(K+2.)*M
      TRAND(16,1)=-4.5*M
      TRAND(16,2)=-1*(2.+K/2.)*M
      TRAND(17,1)=M
      TRAND(17,2)=0.0
      TRAND(18,1)=0.0
      TRAND(18,2)=(K+4.)*M
      TRAND(19,1)=-1.*M
      TRAND(19,2)=0.0
      TRAND(20,1)=0.0
      TRAND(20,2)=-1*TRAND(18,2)
      TRAND(21,1)=2.5*M
      TRAND(21,2)=-1*(M+K*M/2.)
      TRAND(22,1)=M
      TRAND(22,2)=0.0
      TRAND(23,1)=0.0
      TRAND(23,2)=(2.+K)*M
      TRAND(24,1)=-1*M
      TRAND(24,2)=0.0
      TRAND(25,1)=0.0
      TRAND(25,2)=-1*(2.+K)*M
      TRAND(26,1)=-1*M/2.
      TRAND(26,2)=-1*K*M/2.
      TRAND(27,1)=M
      TRAND(27,2)=0.0
      TRAND(28,1)=0.0
      TRAND(28,2)=K*M
      TRAND(29,1)=-1*M
      TRAND(29,2)=0.0
      TRAND(30,1)=0.0
      TRAND(30,2)=-1*K*M
      RESID(1,2)=-1.5*W
      RESID(2,1)=3.*W
```

```
      RESID(2,2)=0.0
      RESID(3,1)=0.0
      RESID(3,2)=W
      RESID(4,2)=0.0
      RESID(5,1)=0.0
      RESID(5,2)=-1*W
      RESID(6,1)=3.*W
      RESID(6,2)=0.0
      RESID(7,1)=0.0
      RESID(7,2)=3.*W
      RESID(8,1)=-3.*W
      RESID(8,2)=0.0
      RESID(9,1)=0.0
      RESID(9,2)=-1*W
      RESID(10,2)=0.0
      RESID(11,1)=0.0
      RESID(11,2)=W
      RESID(12,1)=-3.*W
      RESID(12,2)=0.0
      RESID(13,1)=0.0
      RESID(13,2)=-3.*W
      RESID(14,2)=-1*W/2.
      RESID(15,1)=W
      RESID(15,2)=0.0
      RESID(16,1)=0.0
      RESID(16,2)=W
      RESID(17,1)=-1*W
      RESID(17,2)=0.0
      RESID(18,1)=0.0
      RESID(18,2)=-1*W
      RESID(19,2)=W/2.
      RESID(20,1)=-1*W
      RESID(20,2)=0.0
      RESID(21,1)=0.0
      RESID(21,2)=-1*W
      RESID(22,1)=W
      RESID(22,2)=0.0
      RESID(23,1)=0.0
      RESID(23,2)=W
      PRINT 35
35    FORMAT('IF MORE THAN ONE TRANSISTORS ARE IN THE SAME ISLAND,
     1 OTHER THAN RESISTOR ISLAND PRESS 1 OTHERWISE 2')
      READ *,NST
      IF (NST .NE. 1) GO TO 37
      OPEN(UNIT=11,FILE='inpu22',RECL=80,STATUS='OLD',ACCESS='SEQUENT
     1 IAL',FORM='FORMATTED')
      READ(11,36)(STRAIX(I),STRAIY(I),I=1,5)
36    FORMAT(2F7.2)
      CLOSE(UNIT=11,STATUS='OLD')
37    PRINT 40
40    FORMAT('IF YOU WANT MASKS PRESS 1 OTHERWISE 2')
      READ *,MASK
      PRINT 42
42    FORMAT('KEY IN THE REQUIRED ARRAY SIZE')
      READ *,NARY
      IF (NARY .NE. 1) GO TO 44
      XMIN=0.0
      XMAX=100.0
      YMIN=0.0
      YMAX=100.0
      GO TO 46
44    NA=(100-(NARY/(NARY-1)))/NARY
      NB=NA
46    DO 50 I=1,30
50    TRAND(I,2)=TRAND(I,2)*-1
      CALL GRSTRT(58000,1)
      CALL NEWPAG
      CALL WINDOW(0.,800.,0.,800.)
      DO 60 IN=1,NARY
      DO 60 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 55
```

```
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
55    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL MOVE(BORDRX(1),BORDRY(1))
      DO 60 I=2,N5
      CALL DRAW(BORDRX(I),BORDRY(I))
60    CONTINUE
      IF (NST .NE. 1) GO TO 65
      DO 62 IN=1,NARY
      DO 62 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 61
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
61    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL MOVE(STRAIX(1),STRAIY(1))
      DO 62 I=2,5
      CALL DRAW(STRAIX(I),STRAIY(I))
62    CONTINUE
65    DO 67 IN=1,NARY
      DO 67 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 66
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
66    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL COLLET(N,M)
67    CONTINUE
      DO 70 IN=1,NARY
      DO 70 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 68
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
68    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL MOVE(RILNDX(1),RILNDY(1))
      DO 70 I=2,N4
      CALL DRAW(RILNDX(I),RILNDY(I))
70    CONTINUE
      IF (MASK .NE. 1) GO TO 80
      CALL WAIT(N5,BORDRX,BORDRY,NARY)
80    DO 85 IN=1,NARY
      DO 85 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 83
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
83    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL BASE(N,W,RS,M)
85    CONTINUE
      IF (MASK .NE. 1) GO TO 90
      CALL WAIT(N5,BORDRX,BORDRY,NARY)
90    DO 95 IN=1,NARY
      DO 95 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 93
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
93    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL EMITR(N,M)
95    CONTINUE
      IF (MASK .NE. 1) GO TO 100
      CALL WAIT(N5,BORDRX,BORDRY,NARY)
100   DO 105 IN=1,NARY
```

```
      DO 105 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 103
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
103   CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL METOPN(N,N2,W,RS,M)
105   CONTINUE
      IF (MASK .NE. 1) GO TO 110
      CALL WAIT(N5,BORDRX,BORDRY,NARY)
110   DO 115 IN=1,NARY
      DO 115 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 113
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
113   CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL PAD(N2,M)
115   CONTINUE
      CALL MATXI(N,N1,W,RS,M)
      CALL METLN(N1,N3,M,NARY)
      CALL GRSTOP
      STOP
      END
C     ***************************************************************
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE SUBROUTINE WAIT FOR GETTING MASKS
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE WAIT(N5,BORDRX,BORDRY,NARY)
      DIMENSION BORDRX(20),BORDRY(20)
      CALL CMCLOS
      PRINT 10
10    FORMAT('IF YOU WANT TO GO AHEAD KEY IN 1')
      READ *,NGO
      CALL CMOPEN
      IF (NARY .NE. 1) GO TO 12
      XMIN=0.0
      XMAX=100.0
      YMIN=0.0
      YMAX=100.0
      GO TO 14
12    NA=(100-(NARY/(NARY-1)))/NARY
      NB=NA
14    CALL NEWPAG
      DO 20 IN=1,NARY
      DO 20 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 15
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
15    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      CALL MOVE(BORDRX(1),BORDRY(1))
      DO 20 I=2,N5
      CALL DRAW(BORDRX(I),BORDRY(I))
20    CONTINUE
      RETURN
      END
C     ***************************************************************
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE FOLLOWING IS THE SUBROUTINE TO DRAW COLLECTOR DIFFUSION REGI
C     TRAND1 IS THE LOCALLY USED ARRAY FOR TRANSISTOR DATA.
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE COLLET(N,M)
      INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1 TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2 LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
      REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
      COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
```

```
     1   LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
         DIMENSION TRAND1(5,2)
         REAL M
         DO 60 K=1,N
         GO TO (60,10,50,10),TYPE(K)
 10      CALL MOVE(POSI(K,1),POSI(K,2))
         TRAND1(1,1)=(TRAND(1,1)-(VAL(K)-1.)*2.*M)*SENS(K)
         TRAND1(2,1)=(TRAND(2,1)+(VAL(K)-1.)*4.*M)*SENS(K)
         TRAND1(3,1)=TRAND(3,1)*SENS(K)
         TRAND1(4,1)=(TRAND(4,1)-(VAL(K)-1.)*4.*M)*SENS(K)
         TRAND1(5,1)=TRAND(5,1)*SENS(K)
         DO 15 J=1,5
         TRAND1(J,2)=TRAND(J,2)*SENS(K)
 15      CONTINUE
         IF (ORINT(K) .EQ. 1) GO TO 30
         CALL VECREL
         CALL MOVE(TRAND1(1,1),TRAND1(1,2))
         DO 20 J=2,5
         CALL DRAW(TRAND1(J,1),TRAND1(J,2))
 20      CONTINUE
         CALL VECABS
         GO TO 60
 30      CALL VECREL
         CALL MOVE(TRAND1(1,2),TRAND1(1,1))
         DO 40 J=2,5
         CALL DRAW(TRAND1(J,2),TRAND1(J,1))
 40      CONTINUE
         CALL VECABS
 50      CONTINUE
 60      CONTINUE
         RETURN
         END
C        ****************************************************************
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C        THE FOLLOWING IS THE SUBROUTINE FOR BASE DIFFUSION REGION.
C        TRAND1,RESID1 ARE THE LOCALLY USED ARRAYS FOR TRANSISTOR AND RES
C        OR DATA RESPECTIVELY.
C        ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
         SUBROUTINE BASE(N,W,RS,M)
         INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1   TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2   LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
         REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
         COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1   LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
         DIMENSION TRAND1(5,2),RESID1(13,2)
         REAL L,M
         DO 60 K=1,N
         GO TO (10,20,20,20,30,40),TYPE(K)
 10      L=VAL(K)*W/RS-W
         LL=L/W
         L=LL*W
         RESID(1,1)=-1*(L/2.+3.*W)
         RESID(4,1)=L
         RESID(10,1)=-1*L
         LL=MOD(LL,2)
         DO 15 J=1,13
         RESID1(J,1)=RESID(J,1)*SENS(K)
         RESID1(J,2)=RESID(J,2)*SENS(K)
 15      CONTINUE
         IF (ORINT(K) .EQ. 1) GO TO 17
         IF (LL .EQ. 0) GO TO 12
         POSI(K,1)=POSI(K,1)+W/2.
 12      CALL MOVE(POSI(K,1),POSI(K,2))
         CALL VECREL
         CALL MOVE(RESID1(1,1),RESID1(1,2))
         DO 16 J=2,13
         CALL DRAW(RESID1(J,1),RESID1(J,2))
 16      CONTINUE
         CALL VECABS
         GO TO 60
```

```
17      IF (LL .EQ. 0) GO TO 19
        POST(K,2)=POSI(K,2)+W/2.
19      CALL MOVE(POSI(K,1),POSI(K,2))
        CALL VECREL
        CALL MOVE(RESID1(1,2),RESID1(1,1))
        DO 18 J=2,13
        CALL DRAW(RESID1(J,2),RESID1(J,1))
18      CONTINUE
        CALL VECABS
        GO TO 60
20      CALL MOVE(POSI(K,1),POSI(K,2))
        TRAND1(1,1)=(TRAND(6,1)-(VAL(K)-1.)*2.*M)*SENS(K)
        TRAND1(2,1)=(TRAND(7,1)+(VAL(K)-1.)*4.*M)*SENS(K)
        TRAND1(3,1)=TRAND(8,1)*SENS(K)
        TRAND1(4,1)=(TRAND(9,1)-(VAL(K)-1.)*4.*M)*SENS(K)
        TRAND1(5,1)=TRAND(10,1)*SENS(K)
        DO 25 J=1,5
        TRAND1(J,2)=TRAND(J+5,2)*SENS(K)
25      CONTINUE
        IF (ORINT(K) .EQ. 1) GO TO 27
        CALL VECREL
        CALL MOVE(TRAND1(1,1),TRAND1(1,2))
        DO 26 J=2,5
        CALL DRAW(TRAND1(J,1),TRAND1(J,2))
26      CONTINUE
        CALL VECABS
        GO TO 60
27      CALL VECREL
        CALL MOVE(TRAND1(1,2),TRAND1(1,1))
        DO 28 J=2,5
        CALL DRAW(TRAND1(J,2),TRAND1(J,1))
28      CONTINUE
        CALL VECABS
        GO TO 60
30      CONTINUE
40      CONTINUE
60      CONTINUE
        RETURN
        END
C       ***************************************************************
C       :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C       THE FOLLOWING IS THE SUBROUTINE FOR EMITTER DIFFUSION
C       TRAND1 IS THE LOCALLY USED ARRAY FOR TRANSISTOR DATA.
C       :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        SUBROUTINE EMITR(N,M)
        INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1  TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2  LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
        REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
        COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1  LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
        DIMENSION TRAND1(5,2)
        REAL M
        DO 60 K=1,N
        GO TO (60,10,10,10,30),TYPE(K)
10      NN=VAL(K)
        DO 20 I=1,NN
        CALL MOVE(POSI(K,1),POSI(K,2))
        A=I
        TRAND1(1,1)=(TRAND(11,1)-(VAL(K)-1.)*2.*M+(A-1.)*4.*M)*SENS(K)
        TRAND1(1,2)=TRAND(11,2)*SENS(K)
        DO 15 J=2,5
        TRAND1(J,1)=TRAND(J+10,1)*SENS(K)
        TRAND1(J,2)=TRAND(J+10,2)*SENS(K)
15      CONTINUE
        IF (ORINT(K) .EQ. 1) GO TO 17
        CALL VECREL
        CALL MOVE(TRAND1(1,1),TRAND1(1,2))
        DO 16 J=2,5
        CALL DRAW(TRAND1(J,1),TRAND1(J,2))
16      CONTINUE
```

```
      CALL VECABS
      GO TO 20
17    CALL VECREL
      CALL MOVE(TRAND1(1,2),TRAND1(1,1))
      DO 18 J=2,5
      CALL DRAW(TRAND1(J,2),TRAND1(J,1))
18    CONTINUE
      CALL VECABS
20    CONTINUE
      GO TO 60
30    CONTINUE
60    CONTINUE
      RETURN
      END
C     ***********************************************************
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE FOLLOWING IS THE SUBROUTINE FOR METALISATION OPENING
C     TRAND1,RESID1 ARE THE LOCALLY USED ARRAYS FOR TRANSISTOR AND RE
C     OR DATA RESPECTIVELY.
C     SUBTR IS THE LOCALLY USED ARRAY FOR DRAWING METALISATION OPENIN
C     OF THE SUBTRATE.
C     :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE METOPN(N,N2,W,RS,M1)
      INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
      REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
      COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
      DIMENSION TRAND1(5,2),RESID1(5,2),SUBTR(5,2)
      REAL L,M1
      DO 60 K=1,N
      GO TO (10,20,20,20,50),TYPE(K)
10    L=VAL(K)*W/RS-W
      LL=L/W
      L=LL*W
      RESID(14,1)=-1*(L/2.+2.*W)
      RESID(19,1)=L/2.+2.*W
      M=13
      DO 19 I=1,2
      CALL MOVE(POSI(K,1),POSI(K,2))
      DO 15 J=1,5
      RESID1(J,1)=RESID(J+M,1)*SENS(K)
      RESID1(J,2)=RESID(J+M,2)*SENS(K)
15    CONTINUE
      IF (ORINT(K) .EQ. 1) GO TO 17
      CALL VECREL
      CALL MOVE(RESID1(1,1),RESID1(1,2))
      DO 16 J=2,5
      CALL DRAW(RESID1(J,1),RESID1(J,2))
16    CONTINUE
      CALL VECABS
      GO TO 19
17    CALL VECREL
      CALL MOVE(RESID1(1,2),RESID1(1,1))
      DO 18 J=2,5
      CALL DRAW(RESID1(J,2),RESID1(J,1))
18    CONTINUE
      CALL VECABS
19    M=18
      GO TO 60
20    M=15
      DO 40 I=1,3
      IF (M .NE. 25) GO TO 32
      NN=VAL(K)
      DO 28 I1=1,NN
      A=I1
      TRAND1(1,1)=(TRAND(1+M,1)-(VAL(K)-1.)*2.*M1+(A-1.)*4.*M1)*SENS
     1(K)
      TRAND1(1,2)=TRAND(1+M,2)*SENS(K)
      CALL MOVE(POSI(K,1),POSI(K,2))
```

```
      DO 21 J=2,5
      TRAND1(J,1)=TRAND(J+M,1)*SENS(K)
      TRAND1(J,2)=TRAND(J+M,2)*SENS(K)
21    CONTINUE
      IF (ORINT(K) .EQ. 1) GO TO 26
      CALL VECREL
      CALL MOVE(TRAND1(1,1),TRAND1(1,2))
      DO 25 J=2,5
      CALL DRAW(TRAND1(J,1),TRAND1(J,2))
25    CONTINUE
      CALL VECABS
      GO TO 28
26    CALL VECREL
      CALL MOVE(TRAND1(1,2),TRAND1(1,1))
      DO 27 J=2,5
      CALL DRAW(TRAND1(J,2),TRAND1(J,1))
27    CONTINUE
      CALL VECABS
28    CONTINUE
      GO TO 40
32    DO 33 J=2,5
      TRAND1(J,1)=TRAND(J+M,1)*SENS(K)
      TRAND1(J,2)=TRAND(J+M,2)*SENS(K)
33    CONTINUE
      IF (M .EQ. 20) GO TO 34
      TRAND1(1,1)=(TRAND(1+M,1)-(VAL(K)-1.)*2.*M1)*SENS(K)
      GO TO 35
34    TRAND1(1,1)=(TRAND(1+M,1)+(VAL(K)-1.)*2.*M1)*SENS(K)
35    TRAND1(1,2)=TRAND(1+M,2)*SENS(K)
      CALL MOVE(POST(K,1),POST(K,2))
      IF (ORINT(K) .EQ. 1) GO TO 37
      CALL VECREL
      CALL MOVE(TRAND1(1,1),TRAND1(1,2))
      DO 36 J=2,5
      CALL DRAW(TRAND1(J,1),TRAND1(J,2))
36    CONTINUE
      CALL VECABS
      GO TO 40
37    CALL VECREL
      CALL MOVE(TRAND1(1,2),TRAND1(1,1))
      DO 38 J=2,5
      CALL DRAW(TRAND1(J,2),TRAND1(J,1))
38    CONTINUE
      CALL VECABS
40    M=M+5
      GO TO 60
50    CONTINUE
60    CONTINUE
      SUBTR(1,1)=PADPO(N2,1)-20.
      SUBTR(1,2)=PADPO(N2,2)+20.
      SUBTR(2,1)=SUBTR(1,1)+40.
      SUBTR(2,2)=SUBTR(1,2)
      SUBTR(3,1)=SUBTR(2,1)
      SUBTR(3,2)=SUBTR(2,2)-40.
      SUBTR(4,1)=SUBTR(1,1)
      SUBTR(4,2)=SUBTR(3,2)
      SUBTR(5,1)=SUBTR(1,1)
      SUBTR(5,2)=SUBTR(1,2)
      CALL MOVE(SUBTR(1,1),SUBTR(1,2))
      DO 70 I=2,5
      CALL DRAW(SUBTR(I,1),SUBTR(I,2))
70    CONTINUE
      RETURN
      END
C     ****************************************************************
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE FOLLOWING IS THE SUBROUTINE TO DRAW PADS AND FILL
C     THE MATRIX INFORMATION AND TERMINAL INFORMATION
C     SQRX,SQRY ARE THE LOCALLY USED ARRAYS FOR DRAWING PADS..
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE PAD(N2,M)
```

```
      INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1 TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2 LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
      REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
      COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1 LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
      DIMENSION SQRX(5),SQRY(5)
      REAL M
      DO 20 K=1,N2
      TERG(PADN(K),1)=PADPO(K,2)/M+1
      TERG(PADN(K),2)=PADPO(K,1)/M
      TERG(PADN(K),3)=0
      TERG(PADN(K),4)=0
      I1=TERG(PADN(K),1)-6
      J1=TERG(PADN(K),2)-5
      DO 10 I=J1,J1+11
      MATX(I1,I)=10000000
      MATX(I1+11,I)=10000000
10    CONTINUE
      DO 15 I=I1+1,I1+10
      MATX(I,J1)=10000000
      MATX(I,J1+11)=10000000
15    CONTINUE
      SQRX(1)=PADPO(K,1)-50.0
      SQRY(1)=PADPO(K,2)-50.0
      SQRX(2)=SQRX(1)
      SQRY(2)=SQRY(1)+100.0
      SQRX(3)=SQRX(2)+100.0
      SQRY(3)=SQRY(2)
      SQRX(4)=SQRX(3)
      SQRY(4)=SQRY(1)
      SQRX(5)=SQRX(1)
      SQRY(5)=SQRY(1)
      CALL FILPAN(3,.TRUE.)
      CALL PANEL(5,SQRX,SQRY)
20    CONTINUE
      RETURN
      END
C     **************************************************************
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE FOLLOWING IS THE SUBROUTINE FOR INITIALISATION OF THE
C     ROUTING MATRIX
C     NDIST IS THE LOCALLY USED ARRAY FOR STORING THE LENGTH INFORMAT
C     OF INTERCONNECTIONS FOR SELECTING THE ROUTING PAIRS IN
C     DISJOINT MANNER.
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE MATXI(N,N1,W,RS,M)
      INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1 TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2 LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
      REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
      COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1 LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
      DIMENSION NDIST(50)
      REAL L,M
      DO 70 K=1,N
      GO TO (10,20,20,20,50,60),TYPE(K)
10    L=VAL(K)*W/RS-W
      LL=L/W
      L=LL*W
      IF (ORINT(K) .EQ. 1) GO TO 14
      I=(POSI(K,1)-L/2.-W)/W-1.
      II=(POSI(K,1)+L/2.+2.*W)/W-1.
      M1=(POSI(K,2)-W/2.)/W-1.
      M2=M1
      GO TO 15
14    M1=(POSI(K,2)-L/2.-2.*W)/W-1.
      M2=(POSI(K,2)+L/2.+W)/W-1.
      I=(POSI(K,1)+W/2.)/W-1.
      II=I
15    TERG(TERM1(K),1)=M1+1
```

```
      TERG(TERM1(K),2)=I+1
      TERG(TERM1(K),3)=M1+1
      TERG(TERM1(K),4)=I+1
      TERG(TERM2(K),1)=M2+1
      TERG(TERM2(K),2)=II+1
      TERG(TERM2(K),3)=M2+1
      TERG(TERM2(K),4)=II+1
      DO 17 J1=1,2
      DO 16 J2=M1,M1+2
      DO 16 J3=I,I+2
16    MATX(J2,J3)=10000000
      M1=M2
      I=II
17    CONTINUE
      GO TO 70
20    NN=VAL(K)
      IF (ORINT(K) .EQ. 1) GO TO 32
      IF (SENS(K) .EQ. 1) GO TO 22
      I1=(POSI(K,2)-3.*M)/M-1.
      M1=(POSI(K,1)+M/2.+4.*M+(VAL(K)-1.)*2.*M)/M-1.
      I2=I1+1
      M2=M1-7-(NN-1)*4
      I3=I1+2
      M3=M1-4-(NN-1)*4
      GO TO 24
22    I1=(POSI(K,2)-3.*M)/M-1.
      M1=(POSI(K,1)-M/2.-3.*M-(VAL(K)-1.)*2.*M)/M-1.
      I2=I1+1
      M2=M1+7+(NN-1)*4
      I3=I1+2
      M3=M1+4
24    DO 26 J1=I1,I1+7
      DO 26 J2=M1,M1+2
26    MATX(J1,J2)=10000000
      DO 28 J1=I2,I2+5
      DO 28 J2=M2,M2+2
28    MATX(J1,J2)=10000000
      DO 30 I=1,NN
      M33=M3+(I-1)*4
      DO 30 J1=I3,I3+3
      DO 30 J2=M33,M33+2
30    MATX(J1,J2)=10000000
      TERG(TERM1(K),1)=I1+1
      TERG(TERM1(K),2)=M1+1
      TERG(TERM1(K),3)=I1+6
      TERG(TERM1(K),4)=M1+1
      TERG(TERM2(K),1)=I2+1
      TERG(TERM2(K),2)=M2+1
      TERG(TERM2(K),3)=I2+4
      TERG(TERM2(K),4)=M2+1
      DO 31 I=1,NN
      TERG((TERM3(K)+I-1),1)=I3+1
      TERG((TERM3(K)+I-1),2)=M3+(I-1)*4+1
      TERG((TERM3(K)+I-1),3)=I3+2
      TERG((TERM3(K)+I-1),4)=M3+(I-1)*4+1
31    CONTINUE
      GO TO 70
32    IF (SENS(K) .EQ. -1) GO TO 34
      I1=(POSI(K,2)-M/2.-4.*M-(VAL(K)-1.)*2.*M)/M-1.
      M1=(POSI(K,1)-3.*M)/M
      I2=I1+7+(NN-1)*4
      M2=M1+1
      I3=I1+4
      M3=M1+2
      GO TO 36
34    I1=(POSI(K,2)+M/2.+3.*M+(VAL(K)-1.)*2.*M)/M-1.
      M1=(POSI(K,1)-3.*M)/M
      I2=I1-7-(NN-1)*4
      M2=M1+1
      I3=I2+3
      M3=M1+2
```

```
36      DO 38 J1=I1,I1+2
        DO 38 J2=M1,M1+7
38      MATX(J1,J2)=10000000
        DO 40 J1=I2,I2+2
        DO 40 J2=M2,M2+5
40      MATX(J1,J2)=10000000
        DO 42 I=1,NN
        I33=I3+(I-1)*4
        DO 42 J1=I33,I33+3
        DO 42 J2=M3,M3+2
42      MATX(J1,J2)=10000000
        TERG(TERM1(K),1)=I1+1
        TERG(TERM1(K),2)=M1+1
        TERG(TERM1(K),3)=I1+1
        TERG(TERM1(K),4)=M1+6
        TERG(TERM2(K),1)=I2+1
        TERG(TERM2(K),2)=M2+1
        TERG(TERM2(K),3)=I2+1
        TERG(TERM2(K),4)=M2+4
        DO 44 I=1,NN
        TERG((TERM3(K)+I-1),1)=I3+(I-1)*4+1
        TERG((TERM3(K)+I-1),2)=M3+1
        TERG((TERM3(K)+I-1),3)=I3+(I-1)*4+1
        TERG((TERM3(K)+I-1),4)=M3+2
44      CONTINUE
        GO TO 70
50      CONTINUE
60      CONTINUE
70      CONTINUE
C       THE FOLLOWING PROGRAM IS TO NAME THE RESISTOR TERMINALS
C       APPROPRIATELY
        DO 90 K=1,N
        IF (TYPE(K) .NE. 1) GO TO 90
        I1=-1
        I2=-1
        DO 80 I=1,N1
        IF (TERM1(K) .NE. CONN(I,1)) GO TO 72
        I1=CONN(I,2)
        GO TO 74
72      IF (TERM1(K) .NE. CONN(I,2)) GO TO 74
        I1=CONN(I,1)
74      IF (TERM2(K) .NE. CONN(I,1)) GO TO 76
        I2=CONN(I,2)
        GO TO 78
76      IF (TERM2(K) .NE. CONN(I,2)) GO TO 78
        I2=CONN(I,1)
78      IF (I1 .EQ. -1) GO TO 80
        IF (I2 .EQ. -1) GO TO 80
        GO TO 82
80      CONTINUE
82      A=IABS(TERG(I1,2)-TERG(TERM1(K),2))+IABS(TERG(I1,1)-TERG(
     1  TERM1(K),1))+IABS(TERG(I2,2)-TERG(TERM2(K),2))+IABS(
     2  TERG(I2,1)-TERG(TERM2(K),1))
        B=IABS(TERG(I1,2)-TERG(TERM2(K),2))+IABS(TERG(I1,1)-TERG(
     1  TERM2(K),1))+IABS(TERG(I2,2)-TERG(TERM1(K),2))+IABS(
     2  TERG(I2,1)-TERG(TERM1(K),1))
        IF (A .LE. B) GO TO 90
        DO 84 I=1,4
        I1=TERG(TERM1(K),I)
        TERG(TERM1(K),I)=TERG(TERM2(K),I)
        TERG(TERM2(K),I)=I1
84      CONTINUE
90      CONTINUE
C       THE FOLLOWING PROGRAM FINDS THE PATH DISTANCE OF INTERCONNECTION
        DO 110 I=1,N1
        NDIST(I)=IABS(TERG(CONN(I,2),2)-TERG(CONN(I,1),2))+IABS(TERG(
     1  CONN(I,2),1)-TERG(CONN(I,1),1))
110     CONTINUE
C       THE FOLLOWING PROGRAM ARRANGES THE DISTANCES IN ASSENDING ORDER
C       THIS IS BECAUSE TO SELECT THE INTERCONNECTION PAIRS IN
C       THE DISJOINT MANNER
```

```
      LIMIT=N1-1
      DO 130 I=1,N1
      DO 120 J=1,LIMIT
      IF (NDIST(J) .LE. NDIST(J+1)) GO TO 120
      NTEMP=NDIST(J)
      NDIST(J)=NDIST(J+1)
      NDIST(J+1)=NTEMP
      NTEMP=CONN(J,1)
      CONN(J,1)=CONN(J+1,1)
      CONN(J+1,1)=NTEMP
      NTEMP=CONN(J,2)
      CONN(J,2)=CONN(J+1,2)
      CONN(J+1,2)=NTEMP
120   CONTINUE
130   CONTINUE
      RETURN
      END
C     ****************************************************************
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE FOLLOWING PROGRAM IS THE SUBROUTINE FOR METALISATION
C     SORX,SORY ARE THE LOCALLY USED ARRAYS FOR FILLING THE METALISATI
C     WINDOWS.
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE METLN(N1,N3,M,NARY)
      INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1 TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2 LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
      REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
      COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1 LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
      DIMENSION SORX(5),SORY(5)
      REAL M
      DO 130 K=1,N1
      K1=1
5     IF (TERG(CONN(K,K1),1) .NE. TERG(CONN(K,K1),3)) GO TO 15
      IF (TERG(CONN(K,K1),2) .NE. TERG(CONN(K,K1),4)) GO TO 30
      DO 10 I=(TERG(CONN(K,K1),1)-1),(TERG(CONN(K,K1),1)+1)
      DO 10 J=(TERG(CONN(K,K1),2)-1),(TERG(CONN(K,K1),2)+1)
      MATX(I,J)=0
10    CONTINUE
      GO TO 50
15    IF (TERG(CONN(K,K1),3) .NE. 0) GO TO 30
      I1=TERG(CONN(K,K1),1)-6
      J1=TERG(CONN(K,K1),2)-5
      DO 20 I=J1,J1+11
      IX=MATX(I1,I)/10000000
      IF (IX .GE. 2) GO TO 18
      MATX(I1,I)=0
18    IX=MATX(I1+11,I)/10000000
      IF (IX .GE. 2) GO TO 20
      MATX(I1+11,I)=0
20    CONTINUE
      DO 25 I=I1+1,I1+10
      IX=MATX(I,J1)/10000000
      IF (IX .GE. 2) GO TO 23
      MATX(I,J1)=0
23    IX=MATX(I,J1+11)/10000000
      IF (IX .GE. 2) GO TO 25
      MATX(I,J1+11)=0
25    CONTINUE
      GO TO 50
30    IF (TERG(CONN(K,K1),1) .NE. TERG(CONN(K,K1),3)) GO TO 40
      I1=TERG(CONN(K,K1),1)-1
      J1=TERG(CONN(K,K1),2)-1
      DO 35 I=I1,I1+2
      DO 35 J=J1,TERG(CONN(K,K1),4)+1
      IX=MATX(I,J)/10000000
      IF (IX .GE. 2) GO TO 35
      MATX(I,J)=0
35    CONTINUE
      MATX(I1+1,J1+1)=0
```

```
      GO TO 50
40    I1=TERG(CONN(K,K1),1)-1
      J1=TERG(CONN(K,K1),2)-1
      DO 45 I=I1,TERG(CONN(K,K1),3)+1
      DO 45 J=J1,J1+2
      IX=MATX(I,J)/10000000
      IF (IX .GE. 2) GO TO 45
      MATX(I,J)=0
45    CONTINUE
      MATX(I1+1,J1+1)=0
50    K1=K1+1
      IF (K1 .EQ. 2) GO TO 5
C     THE FOLLOWING PROGRAM SEGMENT DOES THE ROUTING
      IS=TERG(CONN(K,1),1)
      JS=TERG(CONN(K,1),2)
      ID=TERG(CONN(K,2),1)
      JD=TERG(CONN(K,2),2)
      CALL LEESR(N3,M,IS,JS,ID,JD,NARY)
      K1=1
55    IF (TERG(CONN(K,K1),1) .NE. TERG(CONN(K,K1),3)) GO TO 65
      IF (TERG(CONN(K,K1),2) .NE. TERG(CONN(K,K1),4)) GO TO 80
      DO 60 I=(TERG(CONN(K,K1),1)-1),(TERG(CONN(K,K1),1)+1)
      DO 60 J=(TERG(CONN(K,K1),2)-1),(TERG(CONN(K,K1),2)+1)
60    MATX(I,J)=MATX(I,J)+10000000
      GO TO 100
65    IF (TERG(CONN(K,K1),3) .NE. 0) GO TO 80
      I1=TERG(CONN(K,K1),1)-6
      J1=TERG(CONN(K,K1),2)-5
      DO 70 I=J1,J1+11
      MATX(I1,I)=MATX(I1,I)+10000000
70    MATX(I1+11,I)=MATX(I1+11,I)+10000000
      DO 75 I=I1+1,I1+10
      MATX(I,J1)=MATX(I,J1)+10000000
75    MATX(I,J1+11)=MATX(I,J1+11)+10000000
      GO TO 100
80    IF (TERG(CONN(K,K1),1) .NE. TERG(CONN(K,K1),3)) GO TO 90
      I1=TERG(CONN(K,K1),1)-1
      J1=TERG(CONN(K,K1),2)-1
      DO 85 I=I1,I1+2
      DO 85 J=J1,TERG(CONN(K,K1),4)+1
85    MATX(I,J)=MATX(I,J)+10000000
      GO TO 100
90    I1=TERG(CONN(K,K1),1)-1
      J1=TERG(CONN(K,K1),2)-1
      DO 95 I=I1,TERG(CONN(K,K1),3)+1
      DO 95 J=J1,J1+2
95    MATX(I,J)=MATX(I,J)+10000000
100   IF (NARY .NE. 1) GO TO 105
      XMIN=0.0
      XMAX=100.0
      YMIN=0.0
      YMAX=100.0
      GO TO 107
105   NA=(100-(NARY/(NARY-1)))/NARY
      NB=NA
107   DO 120 IN=1,NARY
      DO 120 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 115
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
115   CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      DO 120 I=TERG(CONN(K,K1),1),TERG(CONN(K,K1),3)
      DO 120 J=TERG(CONN(K,K1),2),TERG(CONN(K,K1),4)
      AA=J
      BB=I
      SORX(1)=AA*M
      SORY(1)=BB*M
      SORX(2)=SORX(1)-M
      SORY(2)=SORY(1)
```

```
      SORX(3)=SORX(2)
      SORY(3)=SORY(2)+M
      SORX(4)=SORX(1)
      SORY(4)=SORY(3)
      SORX(5)=SORX(1)
      SORY(5)=SORY(1)
      CALL FILPAN(2,.TRUE.)
      CALL PANEL(5,SORX,SORY)
120   CONTINUE
      K1=K1+1
      IF (K1 .EQ. 2) GO TO 55
130   CONTINUE
      RETURN
      END
C     ******************************************************************
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C     THE FOLLOWING PROGRAM IS THE SUBROUTINE FOR ROUTING
C     CONTR1 IS THE NUMBER OF CELLS TO BE EXPANDED.
C     CONTR2 IS THE NUMBER OF CELLS VISITED DURING EXPANSION OF CELLS.
C     DESTN IS USED TO CHECK WHETHER DESTINATION IS REACHED.
C     ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
      SUBROUTINE LEESR(N3,M,IS,JS,ID,JD,NARY)
      INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1 TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2 LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
      REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
      COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1 LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
      INTEGER CONTR1,CONTR2,DESTN
      REAL M
      DESTN=0
      LIST1X(1)=JS
      LIST1Y(1)=IS
      CONTR1=1
10    CONTR2=1
      DO 50 I=1,CONTR1
      II=LIST1Y(I)
      JJ=LIST1X(I)+1
      IF (JJ .GT. N3) GO TO 20
      CALL TEST1(I,ID,JD,DESTN,II,JJ,CONTR2)
      IF (DESTN .EQ. 1) GO TO 70
20    II=LIST1Y(I)-1
      JJ=LIST1X(I)
      IF (II .LT. 1) GO TO 30
      CALL TEST1(I,ID,JD,DESTN,II,JJ,CONTR2)
      IF (DESTN .EQ. 1) GO TO 70
30    II=LIST1Y(I)
      JJ=LIST1X(I)-1
      IF (JJ .LT. 1) GO TO 40
      CALL TEST1(I,ID,JD,DESTN,II,JJ,CONTR2)
      IF (DESTN .EQ. 1) GO TO 70
40    II=LIST1Y(I)+1
      JJ=LIST1X(I)
      IF (II .GT. N3) GO TO 50
      CALL TEST1(I,ID,JD,DESTN,II,JJ,CONTR2)
      IF (DESTN .EQ. 1) GO TO 70
50    CONTINUE
      CONTR2=CONTR2-1
      DO 60 I=1,CONTR2
      LIST1X(I)=LIST2X(I)
      LIST1Y(I)=LIST2Y(I)
60    CONTINUE
      CONTR1=CONTR2
      IF (CONTR1 .NE. 0) GO TO 10
      CALL CMCLOS
      PRINT 65
65    FORMAT('NO PATH EXISTS')
      CALL CMOPEN
      GO TO 72
70    CALL RETR(ID,JD,IS,JS,M,NARY)
```

```
72      DO 75 I=1,N3
        DO 75 J=1,N3
        II=MATX(I,J)/10000000
        IF (II .GE. 1) GO TO 75
        MATX(I,J)=0
75      CONTINUE
        RETURN
        END
C       ************************************************************
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C       THE FOLLOWING PROGRAM IS THE SUBROUTINE FOR TEST1
C       OCCPD IS USED TO CHECK WHETHER THE CELL IS PERMANENTLY OCCUPIED.
C       VISITD IS USED TO CHECK WHETHER A CELL IS VISTTED PRIVIOUSLY FRO
C       SOME OTHER PATH DURING EXPANSION OF A PERTICULAR CELL.
C       DESTN IS USED TO CHECK WHETHER DESTINATION IS REACHED.
C       CONTR2 IS THE NUMBER OF CELLS VISITED DURING EXPANSION OF CELLS.
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        SUBROUTINE TEST1(I,ID,JD,DESTN,II,JJ,CONTR2)
        INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1  TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2  LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
        REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
        COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1  LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
        INTEGER OCCPD,VISITD,DESTN,CONTR2
        OCCPD=MATX(II,JJ)/10000000
        IF (OCCPD .GE. 1) GO TO 10
        NX=MATX(II,JJ)/1000000
        VISITD=MOD (NX,10)
        IF (VISITD .EQ. 1) GO TO 10
        MATX(II,JJ)=MATX(II,JJ)+LIST1Y(J)*1000+LIST1X(I)+1000000
        LIST2Y(CONTR2)=II
        LIST2X(CONTR2)=JJ
        CONTR2=CONTR2+1
        IF (II .NE. ID) GO TO 10
        IF (JJ .NE. JD) GO TO 10
        DESTN=1
10      RETURN
        END
C       ************************************************************
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
C       THE FOLLOWING PROGRAM IS THE SUBROUTINE FOR RETRACE
C       SORX,SORY ARE THE LOCALLY USED ARRAYS FOR DRAWING THE INTERCONNE
C       ING PATH.
C       ::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
        SUBROUTINE RETR(ID,JD,IS,JS,M,NARY)
        INTEGER TYPE(30),ORINT(30),SENS(30),TERM1(30),TERM2(30),
     1  TERM3(30),CONN(50,2),PADN(20),TERG(75,4),MATX(100,100),
     2  LIST1X(1000),LIST1Y(1000),LIST2X(1000),LIST2Y(1000)
        REAL VAL(30),POSI(30,2),PADPO(20,2),TRAND(30,2),RESID(25,2)
        COMMON TYPE,ORINT,SENS,TERM1,TERM2,TERM3,CONN,PADN,TERG,MATX,
     1  LIST1X,LIST1Y,LIST2X,LIST2Y,VAL,POSI,PADPO,TRAND,RESID
        DIMENSION SORX(5),SORY(5)
        REAL M
        NCONTR=1
        LIST1X(NCONTR)=JD
        LIST1Y(NCONTR)=ID
        LOCAY=ID
        LOCAX=JD
10      LOCA=MOD (MATX(LOCAY,LOCAX),1000000)
        MATX(LOCAY,LOCAX)=MATX(LOCAY,LOCAX)+10000000
        LOCAY=LOCA/1000
        LOCAX=MOD (LOCA,1000)
        NCONTR=NCONTR+1
        LIST1Y(NCONTR)=LOCAY
        LIST1X(NCONTR)=LOCAX
        IF (LOCAY .NE. IS) GO TO 10
        IF (LOCAX .NE. JS) GO TO 10
        IF (NARY .NE. 1) GO TO 12
        XMIN=0.0
        XMAX=100.0
```

```
      YMIN=0.0
      YMAX=100.0
      GO TO 14
12    NA=(100-(NARY/(NARY-1)))/NARY
      NB=NA
14    DO 40 IN=1,NARY
      DO 40 JN=1,NARY
      IF (NARY .EQ. 1) GO TO 16
      XMIN=(JN-1)*(NA+NARY/(NARY-1))
      XMAX=NA+(JN-1)*(NA+NARY/(NARY-1))
      YMIN=(IN-1)*(NB+NARY/(NARY-1))
      YMAX=NB+(IN-1)*(NB+NARY/(NARY-1))
16    CALL VWPORT(XMIN,XMAX,YMIN,YMAX)
      DO 40 I=2,NCONTR-1
      DO 30 J=LIST1Y(I)-1,LIST1Y(I)+1
      DO 30 K=LIST1X(I)-1,LIST1X(I)+1
      IF (J .NE. LIST1Y(I)) GO TO 18
      IF (K .EQ. LIST1X(I)) GO TO 30
18    IF (J .NE. LIST1Y(I+1)) GO TO 20
      IF (K .EQ. LIST1X(I+1)) GO TO 30
20    MATX(J,K)=10000000
30    CONTINUE
      AA=LIST1X(I)
      BB=LIST1Y(I)
      SORX(1)=AA*M
      SORY(1)=BB*M
      SORX(2)=SORX(1)-M
      SORY(2)=SORY(1)
      SORX(3)=SORX(2)
      SORY(3)=SORY(2)+M
      SORX(4)=SORX(1)
      SORY(4)=SORY(3)
      SORX(5)=SORX(1)
      SORY(5)=SORY(1)
      CALL FILPAN(2,.TRUE.)
      CALL PANEL(5,SORX,SORY)
40    CONTINUE
      RETURN
      END
C     ********************************************************************
```

EE-1986-M-SHI-COM